C&R TECHNOLOGIES®

Sinaps®
User's Manual

A Non-geometric Graphical Interface
for SINDA/FLUINT

Sinaps®
User's Manual

Version 5.6

June 2013

This manual, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. The content of this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Cullimore & Ring Technologies. Cullimore & Ring Technologies assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

Preface

Prepared, distributed, and supported by:

C&R Technologies, Inc.
Boulder Colorado, USA

Phone: (303) 971-0292

Web Site: www.crtech.com

Authors:

S. G. Ring
J. E. Baumann
B. A. Cullimore

CHAPTER 1 Introduction 1

Revision History **2**
Improvements in Version 5.2 **2**
Improvements in Version 5.3 **3**
Improvements in Version 5.4 **5**
Improvements in Version 5.5 **8**
Improvements in Version 5.6 **9**

CHAPTER 2 Getting Started 1

Documentation Conventions **1**
Keyboard Shortcuts **1**
Mouse Operation **2**
Help and Support **3**
User's Manual and Help Files **3**
Training **5**
Technical Support **5**
Contact Information 5

CHAPTER 3 Working Environment 1

Working with Files **1**
Overview of the Sinaps Workspace **2**
Task Panels and Windows **3**
Data Windows **3**
Text Windows **5**
Task Panels **7**
Menus and Toolbars **10**
Sinaps Main Menu Options **10**
Setting Preferences 11
Hover Text 12
Read in a SINDA/FLUINT Input File 12
Find and Replace 12
Toolbars **13**
Customizing the Workspace **15**
Docking Task Panels **15**
Hiding Task Panels **17**
Window Management **17**
Customizing Toolbars **20**

CHAPTER 4 Working with Networks 1

Working with a Network **1**
Creating Comments and Drawing Objects **2**

Creating and Managing Layers **2**
Automatic Layer Generation **3**
Creating and Managing Layers **3**
Setting Current Layer 4
Layer Sets **4**
Creating Thermal/fluid Networks **5**
Network Objects **7**
Network Links **9**
Selecting Network Elements **10**
Editing the Network **11**
Network View Options **13**
Network Zoom Options **13**
Network Overview **13**
Panning through the Network **14**
Link Labels **14**
Network Layout Options **14**
Collections **15**
Network Printing and Export **17**
Input Checking **17**
Visual Preprocessing **17**
Color by Material **18**
Color by Layer **19**
Checking for Created Ducts **19**
Network Input Checking **20**
Highlight Associated Paths for Ties **21**
Customizing of Network Field and Elements **21**
Network Field Properties **21**
Background Images in Networks 22
Customizing Appearance of Network Elements **22**

CHAPTER 5 Creating Models in Sinaps 1

Master Model Components **1**
Creating a New Model **4**
Initializing a Model **4**
Setting Units for the Model 4
Setting Network Element Defaults 6
Creating Submodels 7
Thermal Modeling Elements **9**
Thermal Nodes **9**
To create a Thermal Node 10
Editing a Thermal Node 11
Cloning Nodes 14
Setting Node Defaults 14

Thermal Conductors **14**
To Create a Thermal Conductor 15
Editing Conductors 16
Setting Conductor Defaults 18
Generating Multiple Node and Conductors **18**
Flow Modeling Elements **19**
Working Fluids **19**
Defining Fluid Mixtures **22**
Mixture Rules 23
Mixture Output 23
Fluid Lumps **23**
To create a fluid lump 24
Editing a Lump 27
Multi-lump Editing 34
Changing Fluid Lump Submodel 34
Cloning Lumps 34
Setting Lump Defaults 35
Setting Initial Mass Fractions for Mixtures 35
Fluid Paths **36**
To Create a Path 37
Editing a Path 38
Setting Path Defaults 44
Defining Path Type Specific Data **44**
Tube and STUBE Input 44
LOSS and LOSS2 Input 47
Check and Control Valve Input (CHKVLV and CTLVLV) 48
Pressure Regulator Valves (UPRVLV and DPRVLV) 49
Orifice Device (ORIFICE) 50
Capillary Device (CAPIL) 52
SetFlow Devices (MRFSET and VFRSET) 54
Tabular Device (TABULAR) 55
Pump/Fan Device (PUMP) 57
Turbine Device (TURBINE) 68
Variable Displacement Compressor Device (COMPRESS) 80
Positive Displacement Compressor Device (COMPPD) 91
Null Connector (NULL) 100
Heat Transfer Ties **101**
To Create a Tie 104
To Edit a Tie 104
To Move a Tie 110
Setting Tie Defaults 111
Fluid to Fluid Heat Transfer (FTIES) **111**
To Create and FTIE 111

To Edit and FTIE 111
Twin Data 114
Additional Input 114
To Move and FTIES 115
Interface (IFACE) **115**
To Create an Iface 116
To Edit an Iface 116
Twin Data 120
Additional Input 121
To Move an Iface 121
User-defined Data Tables **121**
Table Manager **121**
Paste Text Table **122**
Creating Tables **123**
Conversion and Scaling 126
Additional Text 127
Table Comment 127
Importing and Exporting Tables 128
Referencing Tables (or Array Data) in Logic or Expressions **128**
User-defined Logic **128**
Fortran and Compiling **129**
Specifying Logical Instructions **130**
Column Restrictions 131
Some Simple Statements 132
Simple Repetitions 133
Referring to Processor Variables **133**
Interactions with Registers and Expressions **134**
Global Logic Blocks **134**
Submodel-specific Logic Blocks **135**
User-defined Subroutines **136**
Network Element Logic **137**
Indirect Referencing in Network Logic 140
Network Logic Include Files 142
Thermophysical Property Manager **142**
Object Properties **146**
Editing Network Properties **147**
Editing Network Element Input Parameters **147**
Editing the Visual Appearance of Network Objects and Links **148**
Editing Network Graphical Objects **148**
Editing Post-processing Objects **148**
Editing Post-processing Color Scales **149**
Using INCLUDE Files **149**
Documenting Models **149**

CHAPTER 6 Parameterizing a Model 1

Defining Registers **2**
Importing/Exporting Registers **5**
Register Groups **5**
Using Registers and Expressions **6**
Conditional Operators **6**
Case-specific Registers **7**
Processor Variables **8**
Valid Expressions **10**
Dynamic Registers **10**

CHAPTER 7 Element Generation and Component Models 1

Macros: Element Generation and Component Models **1**
Capillary Evaporator Pumps (CAPPMP) **1**
Creating a Capillary Pump 2
Editing a Capillary Pump 3
Continuous Flow Passages, Pipes, Ducts, and Macros **5**
Automatic Duct Generation **6**
Excluding Specific Elements From Duct Generation 7
Disabling Automatic Duct Generation 7
Pipes **8**
Types of Pipes **8**
Creating a Pipe **9**
Editing a Pipe **10**
Editing Multiple Pipes **23**
Moving Pipes **24**
Connecting Elements Within Pipes **24**
Copying Pipes **24**
Connecting to Pipes **24**
Pipe Orientation **24**
Selecting Network Elements in a Pipe **25**
Relabeling a Pipe **25**
Renumbering a Pipe **25**
Customizing a Pipe **26**
Multi-conductor Element **27**
Creating a Multi-conductor **28**
Editing a Multi-conductor **29**
Post-processing a Multi-conductor **31**

CHAPTER 8 The Model Browser 1

The Model Tree **2**
Using the Submodel Tree **2**

Browsing Pipes and Other Macros **4**
Setting Network Element Defaults **5**
Specifying Fluids **7**
Network diagram **7**
Registers **7**
Tables **8**
Automatic Duct Tree **9**

CHAPTER 9 Using the Case Manager 1

Managing Analysis Cases **1**
Defining Cases **2**
Setting Case Data **4**
Case Data - General **4**
Case Data - Solution **6**
Case Data - Output **10**
Case Data - Build **10**
Case Data - Advanced Options **13**
Options Data 15
Include Paths 17
Registers 18
Tables 18
Global Control Data 19
Global User Data 21
Fluid Property Data 21
Operations 22
Global Logic 24
The Solver 24
Reliability 27
Subroutines 30
Network Includes 30
Submodel-specific Control Data 31
Carray Data 32
Source Data 33
User Logic Blocks 33

CHAPTER 10 Output and Post-processing 1

Controlling Output Frequency and Location **1**
Text Output Files **3**
Standard Text Output **3**
Optional Output Routines **4**
User-defined Output Files **6**
Binary Output for Postprocessing **7**
Creating Binary Output (SAVE file or CSR folder) **8**

Post-processing Operations **9**
Case Manager Post-processing Tab **10**
CSR Folder/Save File Manager 11
Selecting Data to Post-process 13
Coloring and Thickening a Network Diagram **15**
Customizing Post-processing Scales 17
Post-processing Tools: Meters, Gauges and Thermometers **18**
Creating a Post-processing Tool 19
Editing a Post-processing Tool 19
Customizing the Scales for Post-processing Tools 20
Animation **21**
Text (Excel) Post-processing **22**
Extended Text 22
Choked Flow 23
XY Plots **23**
Balance Bar Plots **24**

CHAPTER 11 Export and Import 1

Exporting/Importing Models **1**
Sinaps® vs. SinapsPlus® **1**
Exporting and Importing from SinapsPlus 1
Importing SINDA/FLUINT Text Files **3**
INCLUDE/INSERT File Usage and Restrictions 4
Using CSTARTINC and CENDINC 4
Submodel Import and Export **5**
Table Import and Export **6**
Register Import and Export **6**

CHAPTER 12 Tutorials 1

Creating a Model Template **4**
Customizing the Work Space **8**
Cryogenic Tank Support - Basic SINDA Model **15**
Cryogenic Tank Support - Transient Variation **29**
Cryogenic Tank Support - Temperature-dependent Conductivity Variation **34**
Cryogenic Tank Support - Parametric Sweep **40**
Pipe Flow: Basic Fluid Flow in a Pipe **44**
Filling Scuba Tanks **58**
Model Description **59**

ADDENDUM A Heritage Turbomachinery 1

Heritage Pump/Fan Form (PUMP) **1**

Heritage Turbine Device Form **6**
Heritage Variable Displacement Compressor Form **8**
Heritage Positive Displacement Compressor Form **11**

Index 1

CHAPTER 1

Introduction

Sinaps®, the Sinda Application Programming System, is a graphical user interface for developing SINDA/FLUINT models and for viewing or reporting the results. Due to its graphical nature, the Sinaps work environment makes model preparation and interpretation of the solution results faster and easier than the traditional text file methods. This in turn shortens the training time and significantly reduces modeling errors. Thermal/fluid networks are sketched on the screen using menus and fill-in forms to quickly build and maintain complex models. Models can be validated, runs can be launched, and results can be displayed directly on the network sketch via coloring and other post-processing operations, all within the Sinaps working environment.

Sinaps is essentially a computer-aided design (CAD) system developed specifically to support SINDA/FLUINT. Sinaps is distinguished from Thermal Desktop®, in that it is a *non-geometric* interface compared to Thermal Desktop's *geometry-based* or CAD-based interface. With Sinaps, the user graphically builds an R-C style network by creating node/lump icons at arbitrary locations on the diagram window, and then connects these icons with the appropriate links for transferring heat, mass, and energy.

Sinaps makes it possible to easily maintain, document, and inherit large models. The ability to graphically visualize models and their results simplifies the validation and maintenance processes. The graphical visualization is more readily understood by new analysts, supervisors and customers. Modifying a model is faster and less error prone than with traditional ASCII file input methods.

Sinaps also offers the following capabilities that are not available in SINDA/FLUINT.

- the ability to visualize SINDA/FLUINT results on an arbitrary schematic used to create the input file,
- graphical post-processing on the network diagram by coloring or thickening various objects,
- direct access to XY plotting,
- advanced post-processing features including animation through time, and node/lump energy balance plots,
- customization of the individual elements within a FLUINT LINE or HX macrocommand, or “duct macro,” and direct control over the generation of pipe thermal wall models (for HX macros).
- a Pipe modeling object allows the rapid generation of FLUINT LINE and HX macros in addition to automatic generation of the thermal network including optional outside insulation
- user-defined libraries for material properties and pipes
- the ability to define and re-use sub-networks (user-defined components), and

- a Case Manager to organize multiple variations of a model, and to control their execution.

Other important features of Sinaps include:

- Tools that assist in the depiction of large, complicated networks including poly-line (bent) connections, collection points (collapsed subdiagrams), cloned icons (e.g., multiple instances of the same node or lump), user-sizeable icons, icon alignment, user comment fields, and superimposable layers with individually-controlled visibility.
- Interactive utilities that read-in existing SINDA/FLUINT text input files into Sinaps and display them graphically.

SINDA/FLUINT frees the user from the need for specific, detailed geometry. Before Sinaps, the price for this freedom was the lack of a modern graphical interface. Sinaps provides such graphical access without losing the advantages of the geometry-independent approach. In fact, via the use of algebraic inputs to define model parameters, the benefits of geometry-free modeling can truly be exploited using Sinaps.

1.1 Revision History

Version Number	Type	Date	Link to Summary of Update
Version 5.1	Initial Release	December 2007	
Version 5.2	Annual Update	October 2008	Improvements in Version 5.2
Version 5.3	Annual Update	January 2010	Improvements in Version 5.3
Version 5.4	Annual Update	February 2011	Improvements in Version 5.4
Version 5.5	Annual Update	February 2012	Improvements in Version 5.5
Version 5.6	Annual Update	June 2013	Improvements in Version 5.6

1.1.1 Improvements in Version 5.2

For the benefit of users of Sinaps Version 5.1, features new to Version 5.2 are summarized below:

1. A tabbed option has been added to the Windows menu. This choice is distinguished from that of the Set Preferences option in that it does not affect how the windows will appear next time the model is opened: its election is not saved along with the model.
2. The View tool bar has been expanded to include easy access to the Network and Registers windows.
3. More text output routine options (e.g., LMXTAB, CHOKETAB, etc.) have been added to the Case Manager.
4. An “Extended Text” post-processing option has been added to the network-based Post-processing menu. This option provides on-demand production of Excel-based tabulations roughly equivalent to LMPTAB, PTHTAB, and TIETAB. See “Extended Text” on page 10-22 for details.
5. A major new tool, ***network element logic***, has been provided (Section 5.6.8). This hybrid between data field expressions and Fortran-based user logic blocks provides the best features of both and eliminates most of the difficulties of each. Because user logic-based customizations are very frequent in fluid modeling (e.g., correlation routine calls), network element logic presents a significant enhancement. It simplifies the asso-

ciation between a network object (e.g., a path) and its logic (e.g., a complex K-factor calculation). It can also act as "logic generator" since it allows logic written for one network object to apply simultaneously to many such objects.

6. The ability to export and import submodels has been added. This feature exports/imports both network and network element logic for a submodel. See Section 11.2 for details.
7. Added a new post-processing parameter in the Case Manager to display choked flow.
8. Added a shift key feature to assist in drawing link in the network. See Section 4.3.2 for details.
9. Add the ability to move objects out of a collection by holding the ctrl-alt keys while moving the object.
10. Modified the starting column for inputs in the "additional inputs" block for fluid elements. Previously contents started in column 1, they now start in column 2. This aids in preventing preprocessor errors.
11. Added ability to use an underscore character '_' in a submodel name.

In SINDA/FLUINT Version 5.1 and earlier, the default solution method for thermal submodels was MATMET=0, meaning that purely iterative solutions were used. Such iterative methods are inefficient for very large thermal submodels typical of Thermal Desktop modeling. Most Sinaps models use only modest thermal networks and almost always have fluid submodels, but this combination often caused thermal models to require extra solution iterations that were not required by FLUINT, which solves equations simultaneously and usually in far fewer iterations.

In Version 5.2 and later, the default is now MATMET=12 for new models,[1] meaning a single simultaneously-solved matrix is used for thermal submodels. This often means far fewer overall iterations. However, it also means that errors in thermal logic or networks might not be as tolerable, and might be harder to isolate. If problems arise, try MATMET=0 and refer to the SINDA/FLUINT User's Manual for full details.

1.1.2 Improvements in Version 5.3

For the benefit of users of Sinaps Version 5.2, features new to Version 5.3 are summarized below:

Extended Macros and Autogeneration of Ducts (SINDA/FLUINT Macros). The automatic aggregation of similar elements in continuous flow passages has been updated to support changes in SINDA/FLUINT Version 5.3. These changes now allow HX and Line macros to be a collection of *both* junctions and tanks and *both* tubes and stubes (previously macros were limited to only one lump type and one path type). This new feature allows the aggregation of elements on a larger scale to create "extended macros." These extended macros (referred to as ***Ducts*** in Sinaps) may contain individual lumps and paths but may also contain one or more pipes. This allows the creation of one continuous flow passage through a piping network as long as the inlet and outlet flow areas of adjacent paths are similar and appear to be the primary flow passage.

The duct concept provides the ability model single flow passage with varying diameters along its length as multiple pipes while retaining the ability to automatically calculate spatial accelerations throughout the duct due to heating, area changes, or side flow. Previously each pipe was constituted a separate duct. To from a duct, two pipes of dissimilar flow areas, must be connected with a flow reducer or expander (tube/stube or a pipe with unique inlet and outlet flow areas defined, i.e. AFI/AFJ). See "Automatic Duct Generation" on page 7-6 for more information.

1 Older models will be read into Version 5.2 with whatever MATMET settings were used in those models. Consider converting them to MATMET=12 if they tolerate that solution method.

Expanded Pipe Capabilities. The pipe object has been redesigned in version 5.3. The primary purpose of the redesign was to expand capabilities such as adding automatic calculation of thermal capacitance and conductance for the pipe wall, ability to add insulation to a pipe, the addition of built-in standard pipe tables, and the ability to create a user definable pipe library for custom pipes. See "Pipes" on page 7-8 for more details.

A secondary purpose in the redesign was to allow more flexibility with regard to customizing the objects within the pipe without locking in pipe level parameters. For example, previously setting the FK factor on a path in a pipe resulted in the entire pipe being customized and the subsequent inability to change the lump type or path type for the pipe. The pipe form now controls *only* the parameters defined within the pipe form such as geometric definition, element types, resolution, insulation, etc. All element level parameters not defined via the pipe form (path K factors, lump compliance, etc.) become the responsibility of the user to define outside of the pipe form.

The new pipe dialog provides the following expanded capabilities:

1. New options allow flow cross sections as circular, rectangular, and user defined.
2. The wall roughness fraction parameter (WRF) is now defined at the pipe level instead of the path level. The WRF value will be applied to all paths within the pipe.
3. Automatic calculation of pipe wall thermal capacitance and conductances.
4. Ability to define pipes with two radial nodes in the pipe wall.
5. Multiple wall cross sections including circular, rectangular, and user defined. Both circular and rectangular have options to define the pipe as a mass/length or as a cross section.
6. Built-in tables for standard pipe schedules and sizes for user with circular pipes.
7. Ability to add insulation to the exterior of a pipe. Up to ten layers of insulation can be defined, each with a unique thickness and material.
8. New fill/purge options have been added to a pipe to assist in flat front modeling.
9. New user definable pipe library. The library saves pipe parameters such as diameter, flow areas, wall roughness, insulation, etc. Pipe length, resolution and duplication factors are not saved as part of the library. The resulting library can be accessed from any Sinaps model.
10. New multi-conductor network element can be used to generate a series of conductors connecting the outer surface of a pipe to a single node, perhaps as a convective or radiative environmental connection.

Pipes from versions prior to Sinaps 5.2, text file import, or Sinaps*Plus* import will migrate as "user defined" cross section, with a "user defined" wall geometry. Migrated pipes will need to be edited, defining the cross section and wall geometry along with associated parameters, in order to take advantage of the new pipe features.

Miscellaneous Improvements.

1. New ability added to Model Browser to view automatically generated macros.
2. New right click option on pipes to allow element level editing.
3. New option added to fluid lumps to allow a lump to not be aggregated into automatic ducts.
4. Support for new *global* level logic. Note that output now defaults to the global output block.
5. New ability to defined registers as double precision.
6. Added ability to define temperature dependent thermophysical properties using either absolute or standard temperature units. Previously property tables were required to be in absolute temperature units.
7. Revised the pipe renumbering form to allow renumber of twins.

8. Added ability to create selection sets through the model browser.
9. Added new ability to select multiple pipes in the model browser for top level editing. This feature is useful for toggling element types in pipes, see “Browsing Pipes and Other Macros” on page 8-4.
10. Added support for expanded names in SINDA/FLUINT version 5.3. Submodel names, register names, and build configuration names can now be 32 characters long. Expressions can be 1000 characters long. Note that Fortran logic sections are still limited to 132 characters due to the limit in most Fortran compilers.
11. EZXY plotter updated for improved compatibility with Thermal Desktop. Note that 5.1 and early 5.2 plot files may need to be migrated through a later 5.2 version before they can be opened in 5.3.
12. To provide compatibility with the new SINDA/FLUINT feature “Functions and Subroutine Interfaces,” the reserved word list has expanded to include all SINDA/FLUINT subroutine names and more. Older models which previously ran may now throw a preprocessor error. For example if a model contained a register named “Tsink” it will result in a preprocessor error since the name is in conflict with the subroutine named TSINK.

1.1.3 Improvements in Version 5.4

For the benefit of users of Sinaps Version 5.3, features new to Version 5.4 are summarized below:

Tables. A new feature referred to as ***Tables*** has been introduced with version 5.4. Tables provide a familiar grid structure for inputting columnated or tabular data which are subsequently converted to traditional SINDA/FLUINT arrays when Sinaps generates the SINDA/FLUINT input file. The grid structure allows data to be pasted from other applications including Microsoft Excel™. Tables can be scaled using registers or expressions and are identified by descriptive user definable names such as “PDU Heat Dissipation”. A ***Table Manager*** provides access to all tables and is accessible through the Model Browser. The Table Manager replaces the previous Array Data which was accessible through the advanced tab in the Case Manager.

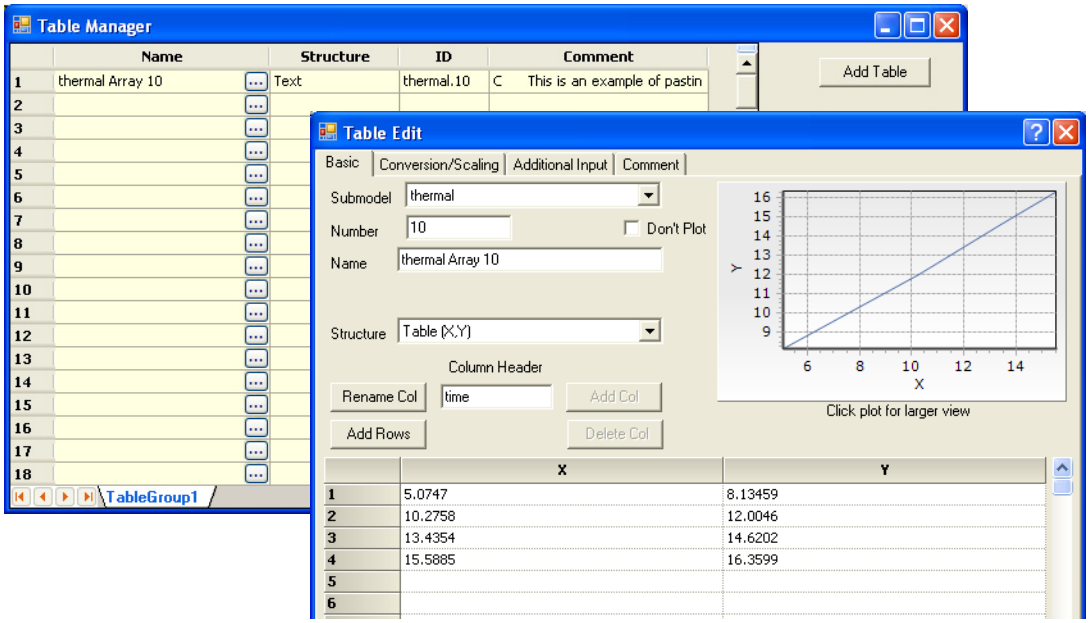


Table formats include a single column table which creates a singlet SINDA/FLUINT array, a XY table creating a doublet array, or an XY map which creates a bivariate array. A text based table allows definition of less common arrays structures such as a trivariate and supports importing of text based models.

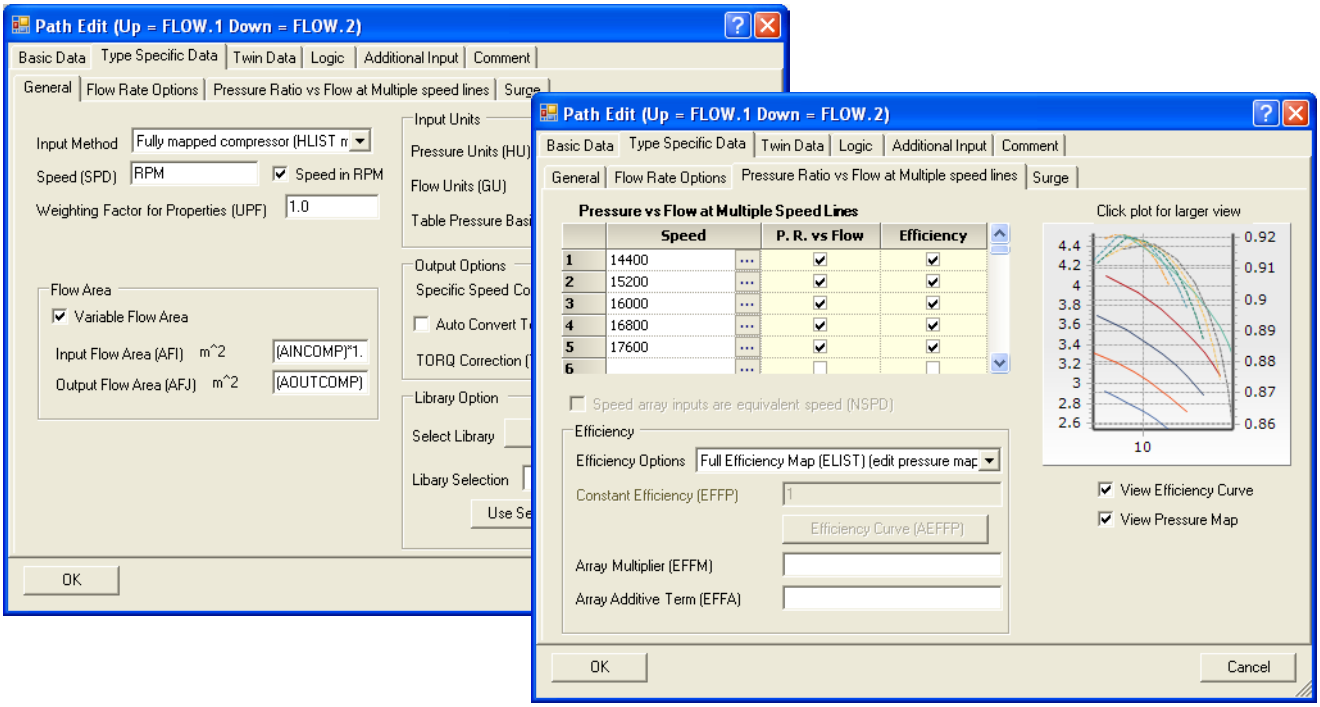
Tables can be referenced directly from network objects. For example if a SIV node is being created, instead of providing an array number for the temperature varying capacitance, the node edit form provides the ability to browse the table manager and select the appropriate table. Tables are assigned numbers when created so they can be referenced from user logic just like traditional arrays (A1, NA1,submodel.A1, etc.). Tables are used extensively by the new turbomachinery forms for defining performance maps.

When migrating a 5.3 or earlier model or importing from a text file or Sinaps*Plus,* arrays will be imported as text based tables. The user has the option of individually converting text based tables to any standard table format. When importing, Sinaps will read the arrays in the first case in as the "base" tables to be placed in the Table Manager. As it goes through the other cases, any *case specific* arrays will be defined as table overrides on the advanced tab of the case manager. Any *new* arrays found in cases will be added to the main Table Manager.

For detailed information on tables please see “User-defined Data Tables” on page 5-121

Redesign of Turbomachinery Forms. The input forms for pumps, compressors, and turbines have been redesigned to provide a more user-friendly interface and assist the user in determining which inputs are required. For example, when creating a pump the user has the option of creating a simple pump based on a single performance curve or a fully mapped pump. Depending on which type of pump is selected, the contents of the form will change to only display appropriate input options.

When creating a turbomachinery device in Sinaps 5.4, all traditional array inputs are now defined using tables. The table dialog has been integrated directly into the pump input form for ease of access and direct plotting of curves.



New Pool Boiling (HTP) Ties. A new fluid tie type has been added to support the addition of Pool Boiling ties in SINDA/FLUINT. These ties automate the calculations surrounding pool boiling, including estimation of post critical heat flux operation, and suppression or reduction of boiling deep beneath the surface of the "pool".

For in depth information on the Pool Boiling ties, see section 3.6.3.3 of the SINDA/FLUINT Users' Manual. To learn more about how use define these ties in Sinaps, see "Heat Transfer Ties" on page 5-101.

New Controllable Orifices. The fluid orifice device has been expanded to include a controllable mode (MODA) to provide the ability to simulate the opening or closing of a valve, a backpressure (upstream pressure) regulator, or a (downstream) pressure regulator. When the orifice is set in "control mode" (positive MODA), the user can set upper and lower limits on the orifice size (AORI), a maximum rate of change and a time constant for the valve. In pressure regulating mode (negative MODA), the user defines a pressure set point and FLUINT will determine the best value for AORI.

For in depth information on controllable s, see section 3.5.10.2 of the SINDA/FLUINT Users' Manual. To learn more about how use define these in Sinaps, see "Orifice Device (ORIFICE)" on page 5-50.

Miscellaneous Improvements.

1. Built-in water description has been changed. Due to limitations in the SINDA/FLUINT built-in water properties, FID=718 (simple water) is no longer available. If water is selected as the working fluid, Sinaps use the include file f6070_water.inc which is a full two-phase, incompressible liquid description. For existing models which used the built-in water description, Sinaps will automatically update the model to point to this file.

2. For consistency, FloCAD and Sinaps have both been updated to use the same built-in air description, air8_1atm.inc. All existing models which previously used the built-in air description will automatically be converted. This new air description is similar to the previous air description but has a upper temperature limit. The lower limit of the description has been increase from 90K to 132.5 (critical temperature).
3. Sinaps has been updated to support SINDA/FLUINT's transition to being full 64-bit double precision internally. All data with the exception of the thermophysical properties file (*.tdp) is now treated as double precision during execution. Register types "float" and "double" are now identical, both will create a double precision register. All data on save files is also now double precision.
4. Please note that new words have been added to the reserved word list, a full list is available in section 6.18.5 of the SINDA/FLUINT User's Manual. New reserved words are CSF, SPR, `THKL, ACCM, DEPTH,` MODA, AORI_L, AORI_H, AORI_C, and AORI_T. Current models using these names as registers, constraint variables, etc. will need to be revised to use new names.

1.1.4 Improvements in Version 5.5

For the benefit of users of Sinaps Version 5.4, features new to Version 5.5 are summarized below:

Export and Import of Tables. A new feature has been added to allow the export and import of tables. Tables can be exported individually, as a selected subset of tables, or as a whole (all tables). Once exported tables can be imported into other models or perhaps cases.

Expanded Input Parameters for Twinned Tanks. The tank edit form has been expanded to include commonly used parameters associated with defining twinned tanks, such as interface surface area (AST) and heat transfer (ULT and UVT) along with droplet and bubble velocities (VDRP and VBUB).

Miscellaneous Updates to Support SINDA/FLUINT Version 5.5.

1. Void fraction (AL) is now an input for fluid lumps. If AL is specified, the initial XL (vapor mass fraction) is ignored. For twinned tanks, specifying AL corresponds to setting the initial volume of each tank: AL*VOL for the secondary/vapor tank, and (1.0-AL)*VOL for the primary/liquid tank. This change will not impact existing models but note that new models will default to using AL instead of XL, however the user can override this default.
2. New option for initializing the temperature (TLV) of the secondary tank (vapor/ullage) in a twinned tank This option signals the twins to start in the split state, even if the phasic temperatures are the same initially.
3. Added the ability to suppress the writing out of built-in constants as registers.
4. Added a new option in the network Input Checks menu to easily access and set the parameters for network input coloring and hover text.
5. Added toolbar access for Sinaps Tables.
6. New reserved words include ATEST_SP through ZTEST_SP which are now named single precision user constants for use in logic.
7. Please see the EZXY User's Manual for 5.5 plotter updates.
8. Please see the SINDA/FLUINT User's Manual Revision History for additional SINDA/FLUINT related updates.

1.1.5 Improvements in Version 5.6

For the benefit of users of Sinaps Version 5.5, features new to Version 5.6 are summarized below:

Support for Compressed Solution Results (CSR Folders). SINDA/FLUINT provides a new alternative to the traditional binary SAVE file for post-processing and restarts. Unlike the single binary SAVEfile which was not conducive to allowing the user access to the data with third party software, the new Compressed Solution Results (CSR) folder is formatted to allow user access. Other advantages of the new CSR folder are smaller size on disk, and faster execution time for large models. Small models will most likely see an increase in execution time.

The default for new models continues to be the SAVE file option, however this will probably change to the CSR folder option in future releases. When opening an existing 5.5 Sinaps model in 5.6, Sinaps will retain the save file usage. To migrate a model to use the CSR option, use the menu option `Model>User CSR` and click to enable the option (a check mark will appear to the left of the menu item if is enabled). Users are encouraged to become familiar with the CSR option and consider migrating models to the CSR option since the save file option will become obsolete and unsupported in the future.

Twinned Tank and Flat Front Expansions. Sinaps has been updated to support the twinned tank expansions made in SINDA/FLUINT to better handle boiling and condensation within twinned tanks. The twinned tank edit form now supports the expanded options for VDRP and VBUB.

SINDA/FLUINT has been revised with version 5.6 to better handle flat front modeling. Although Sinaps has always supported flat front (fill/purge) modeling through the Pipe edit form, some changes have been made within Sinaps related to how these Fill/Purge pipes can be aggregated into ducts.

- In prior versions of Sinaps, only centered pipes could be aggregated into ducts, i.e. multiple centered pipes and individually input lumps and paths could be aggregated into a continuous duct. Upstream/downstream and fill/purge pipes would not aggregate into a duct. Starting with version 5.6, Sinaps will now aggregate “like” pipes and individual network elements, i.e. upstream pipes can be aggregated with other upstream pipes but not with centered, downstream, fill, or purge pipes. This aggregation process can pull individually input lump and paths and pipes into upstream, downstream, fill, or purge, ducts.
- Improper aggregation will produce erroneous results. Due to the complexity of the aggregation process, it is highly recommended that the user *always* use the `Show Created Ducts` input check to see where ducts have been created and manually adjust the aggregation if needed (ducts can also be reviewed in the Model Browser). The rules of aggregation are very complex and the code can err easily. To help identify incorrect aggregation, in addition to the input check feature within Sinaps, SINDA/FLUINT has been updated and will write out warning messages if improper or questionable aggregation is encountered.

To learn more about duct modeling, please see “Continuous Flow Passages, Pipes, Ducts, and Macros” on page 7-5. To learn more about aggregation in Sinaps, please see “Automatic Duct Generation” on page 7-6.

Miscellaneous Updates to Support SINDA/FLUINT Version 5.6.

1. Support for new indirect referencing options to refer to the secondary (twin) lump, path, tie, or FTIE. For example, the #LUMPTWIN of a tie refers to the secondary tank of the twinned lump to which it is connected.

2. A new directional vector based option has been added for pipe positioning data. This new method is now the preferred method and takes into account changes to the pipe length and pipe discretization methods. See “Pipe Position” on page 7-22.
3. Lump position data for body forces (CX, CY, and CZ) have been added to the list of available parameters for input checking.
4. A new option has been added to the Fluid tab of the Pipe edit form to provide the analyst the option of not having the pipe modeled as a duct. Note that it is highly recommended Pipes always be modeled as a duct (the default). Disabling duct generation for a pipe will disable the automatic calculation of spatial acceleration associated with density and area changes, transient storage, and side passages. If disabled, the user is responsible for adding these calculations through user defined logic.
5. Added a new right click option on heat transfer ties to highlight the associated paths. This provides a user with a visual check on the tie inputs.
6. Added access (middle mouse button click menu) to the Sinaps table manager from logic blocks and the network logic dialog. This allows the user to open the table manager to view table IDs or to add new tables without having to leave the network logic dialog or logic blcok text window.

CHAPTER 2

Getting Started

2.1 Documentation Conventions

Throughout this manual various conventions are used to highlight information. The samples below identify the various conventions.

Notes: Notes are used to highlight cautions and warnings of the user should be aware.

> Note: Do not run with scissors.

Menu Options: Throughout the user's manual there are instructions on how to navigate the various pull down menus. Tiered menu options are separated by a ">"

Example: Save/Show>Show Values

Definitions: When a term is first introduced, it will be emphasized using ***bold italics***

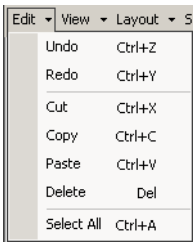
2.2 Keyboard Shortcuts

Sinaps uses standard Windows-based keyboard shortcuts. Keyboard shortcuts provide an alternative method to activate many of the pull-down menu options and are available in text input windows, network diagrams and the various task panels.

Keyboard shortcuts are activated by holding down the *control* (Ctrl) key and the pressing the appropriate keyboard letter. Keyboard shortcuts, if available, are listed to the right of the text found in a given pull-down menu. For example, Figure 2-1 shows the possible keyboard shortcuts available for editing in the network window.

A list of all keyboard shortcuts is not provided in this manual both because they are listed in the menus themselves, and because it is important for the user to first understand how each pull-down option functions before becoming accustomed to a keyboard shortcut.

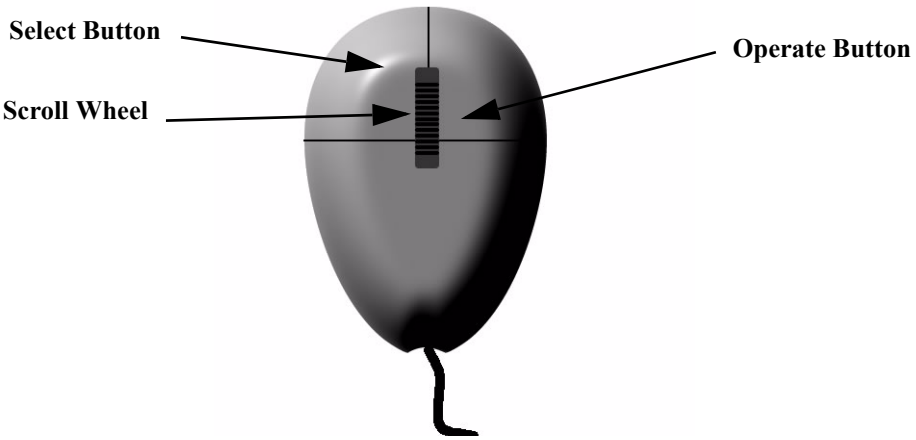
FIGURE 2-1. Keyboard Shortcuts



2.3 Mouse Operation

The computer mouse performs three basic operations: select, operate and window (see Figure 2-2).

FIGURE 2-2. Mouse Buttons



Select Button: Select is used to choose a menu item or icon, to position the text cursor, or to highlight text. This is the primary button used for mouse operations. Once an object or window has been selected this button allows you to drag the item across the screen.On a two- or three-button mouse, the select button is the left button. To select multiple network elements, hold the control key while selecting the elements. Note that holding the shift key when clicking forces the curser into draw mode for drawing links between objects.

Operate Button: The operate button brings up a menu of operations that are appropriate for the current view or selection. (This menu will be different depending on where the cursor is located, and whether or not any items have been selected.) On a two- or three-button mouse, operate is the right-hand button.

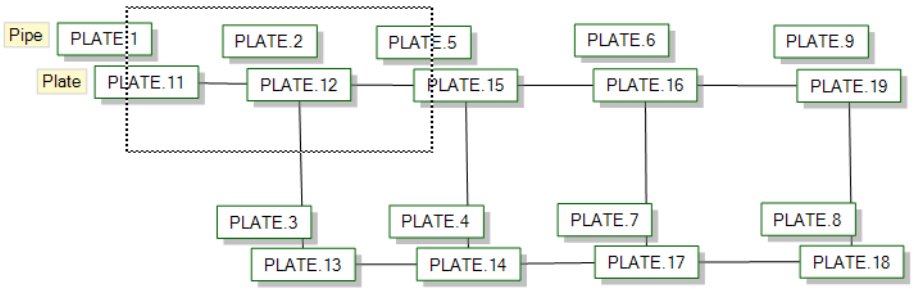
Scroll Wheel/Middle Button: Some mice have scroll wheels which allows you to vertically scroll through windows. This scroll function will work in the various Sinaps windows. On most mice, the scroll wheel also performs as a

middle button (this may need to be programmed). The “middle” button will allows the user access additional content menus in the network and to pan through the model network.

Windowing for Object Selection (Selection by Areal Drag): Windowing can be used to select objects in the network diagram. To window objects simply click the select (left) mouse button and while holding it down, drag the cursor to the opposite corner of the desired box.

Object selection will vary depending on the direction the cursor is dragged. Dragging a window from upper corner to a lower corner will select everything which resides *entirely* in the box. Dragging the window from the lower corner to an upper corner will select everything within *and* everything *intersected* by the line of the box. For example, if drawn from upper left to lower right, the window shown in Figure 2-3 will select PLATE nodes 2 and 12 along with the two conductors to the left and right of node PLATE.12. If drawn from lower right to upper left, the selection will include PLATE nodes 1, 2, 5, 11, 12, 13, and all three conductors attached to node PLATE.12.

FIGURE 2-3. Window Selection Methods



2.4 Help and Support

2.4.1 User’s Manual and Help Files

This User’s Manual, available in PDF format through the start menu, is the content basis for the Sinaps help file system. The User’s Manual and associated Sinaps help files provide explanations of Sinaps and its usage. *No attempt is made to explain SINDA/FLUINT usage. Documentation and training for SINDA/FLUINT usage is available separately.*

Sinaps provides two types of help systems, general and dynamic. The general help is accessible by clicking on the main help menu or help menus in windows such as the network, registers, or clicking the F1 function key in any data window or task panel. The general help will open in a separate window from Sinaps. This help system shown in Figure 2-4 provides access to the table of contents and index along with the ability to search the manual. It also allows the user to tag specific frequently used topics as “favorites” for quick reference.

The dynamic or context-sensitive help is available from most edit forms. When the question mark icon in the upper right corner of any dialog is clicked, the dynamic help will appear on the right side of the screen as shown in Figure 2-5. Selecting the F1 function key from any edit form will also bring up the dynamic help. This help format is context-

sensitive and is associated with the open dialog. When the dialog window is closed the dynamic help window will automatically be dismissed.

FIGURE 2-4. General Help System Example Window

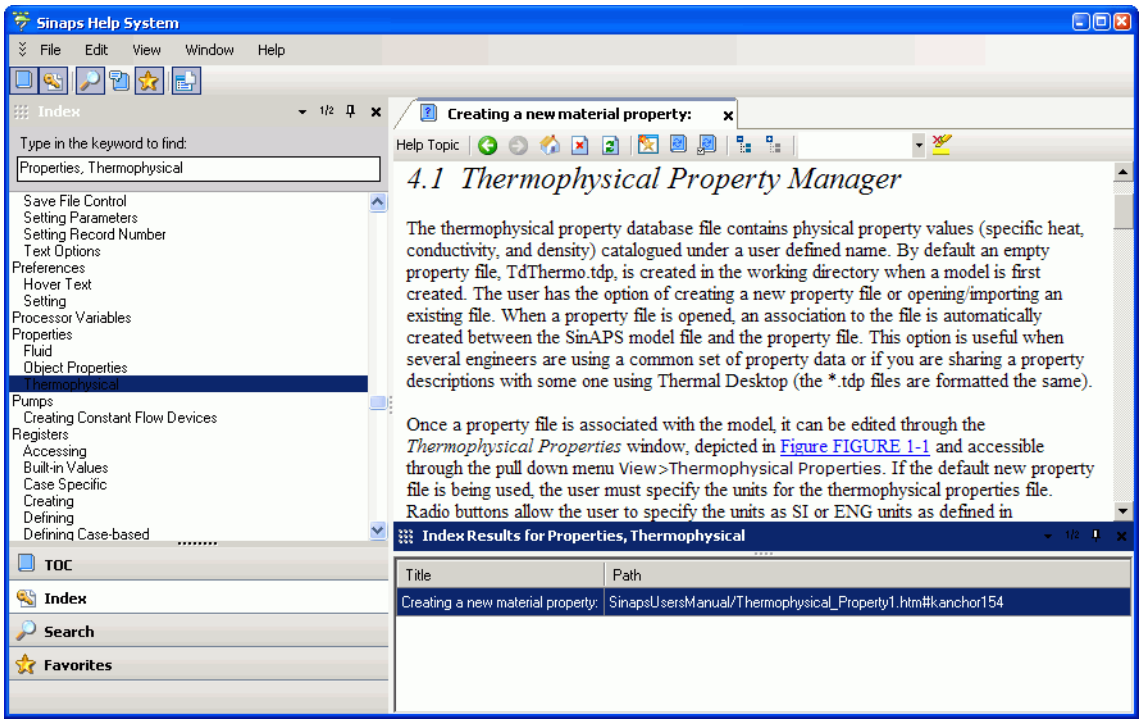
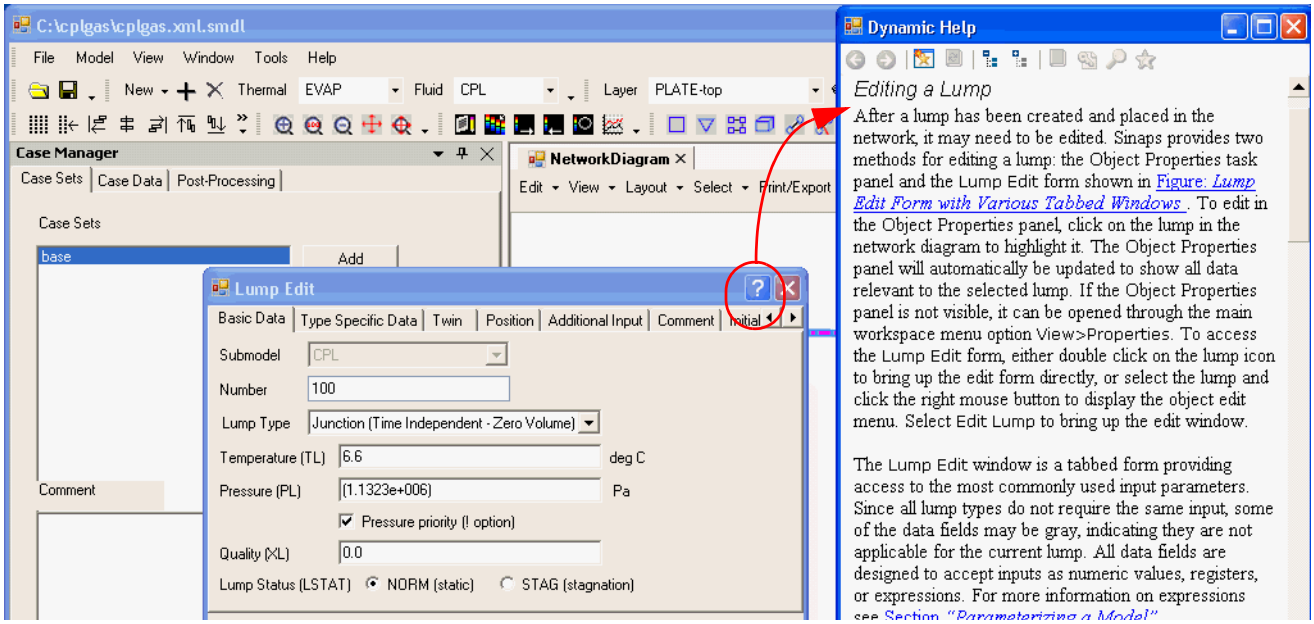

Sinaps Help System
File Edit View Window Help
Index
Type in the keyword to find:
Properties, Thermophysical
Save File Control
Setting Parameters
Setting Record Number
Text Options
Preferences
Hover Text
Setting
Processor Variables
Properties
Fluid
Object Properties
Thermophysical
Pumps
Creating Constant Flow Devices
Registers
Accessing
Built-in Values
Case Specific
Creating
Defining
Defining Case-based
TOC
Index
Search
Favorites
Creating a new material property:
Help Topic
4.1 Thermophysical Property Manager
The thermophysical property database file contains physical property values (specific heat, conductivity, and density) catalogued under a user defined name. By default an empty property file, TdThermo.tdp, is created in the working directory when a model is first created. The user has the option of creating a new property file or opening/importing an existing file. When a property file is opened, an association to the file is automatically created between the SinAPS model file and the property file. This option is useful when several engineers are using a common set of property data or if you are sharing a property descriptions with some one using Thermal Desktop (the *.tdp files are formatted the same).
Once a property file is associated with the model, it can be edited through the Thermophysical Properties window, depicted in Figure FIGURE 1-1 and accessible through the pull down menu View>Thermophysical Properties. If the default new property file is being used, the user must specify the units for the thermophysical properties file. Radio buttons allow the user to specify the units as SI or ENG units as defined in
Index Results for Properties, Thermophysical
Title
Path
Creating a new material property:
SinapsUsersManual/Thermophysical_Property1.htm#kanchor154

FIGURE 2-5. Context-Sensitive Help Window


C:\cplgas\cplgas.xml.smdl
File Model View Window Tools Help
New Thermal EVAP Fluid CPL Layer PLATE-top
Case Manager
Case Sets Case Data Post-Processing
Case Sets
base
Add
Comment
NetworkDiagram
Edit View Layout Select Print/Export
Lump Edit
Basic Data Type Specific Data Twin Position Additional Input Comment Initial
Submodel CPL
Number 100
Lump Type Junction (Time Independent - Zero Volume)
Temperature (TL) 6.6 deg C
Pressure (PL) (1.1323e+006) Pa
Pressure priority (! option)
Quality (XL) 0.0
Lump Status (LSTAT) NORM (static) STAG (stagnation)
Dynamic Help
Editing a Lump
After a lump has been created and placed in the network, it may need to be edited. Sinaps provides two methods for editing a lump: the Object Properties task panel and the Lump Edit form shown in Figure: Lump Edit Form with Various Tabbed Windows. To edit in the Object Properties panel, click on the lump in the network diagram to highlight it. The Object Properties panel will automatically be updated to show all data relevant to the selected lump. If the Object Properties panel is not visible, it can be opened through the main workspace menu option View>Properties. To access the Lump Edit form, either double click on the lump icon to bring up the edit form directly, or select the lump and click the right mouse button to display the object edit menu. Select Edit Lump to bring up the edit window.
The Lump Edit window is a tabbed form providing access to the most commonly used input parameters. Since all lump types do not require the same input, some of the data fields may be gray, indicating they are not applicable for the current lump. All data fields are designed to accept inputs as numeric values, registers, or expressions. For more information on expressions

2.4.2 Training

C&R Technologies, Inc. offer a variety of training opportunities. It is recommended that all new users as a minimum work through the tutorials and review the SINDA/FLUINT Sample Problem Appendix.

- *Tutorials:* Simple tutorials which provide an introduction to software usage are included in this manual. It is recommended all new users work through these tutorials.
- *Sample Problems:* The SINDA/FLUINT Sample Problem Appendix provides a variety of sample problems with complete documentation on model development options and choices. We recommend all users review these sample problems.
- *Introductory training classes:* We offer these classes at various times during the year. These classes are held in the Denver Colorado area. The class schedules are posted on our web site.
- *On-site and custom training classes:* We can provide standard training classes of custom training classes at your facility.
- *Online mini-courses:* We offer a variety of free web-based short courses on specific modeling topics. The topics and schedule for these classes are posted on our web site at www.crtech.com.

2.4.3 Technical Support

Please check the *Online Technical Database* for announcements, software support, and hardware discussions at http://www.crtech.com/forum/index.php. Please check this database for support questions prior to contacting our support department. If your question cannot be answered in through the Technical Database, questions should be submitted through our web site at http://www.crtech.com/forms/contact.html

For software updates and other general information visit our web site at http://www.crtech.com.

2.4.3.1 Contact Information

C&R Technologies, Inc.
Voice: (303) 971-0292
FAX: (303) 971-0035

Web Site: www.crtech.com

CHAPTER 3

Working Environment

Sinaps is a window-based application designed to work within Microsoft's **.**NET framework. The model files created by Sinaps are in XML format. Models from prior versions of Sinaps*Plus*® must be exported as XML before they can be brought into Sinaps. Sinda text-based input decks may also be imported.

Although models created in the Sinaps work environment correspond to an underlying XML-based model file, traditional SINDA/FLUINT text input files are also created and sent to the SINDA/FLUINT processor for execution, or they may be created by the user for review. However, those SINDA/FLUINT text files do not contain graphical information and might have other mark-ups, so they should not be used to archive Sinaps models nor should be used for re-import. Traditional SINDA/FLUINT input files should only be used for reviewing, or for running SINDA/FLUINT.

After executing a case, the Sinaps work space is used to post-process results on the network, through XY plots, and various text files.

3.1 Working with Files

Sinaps models will contain at least one model file in XML format (*.xml.smdl) and possibly a thermophysical properties file (*.tdp). Additional files may include register data and plot files. The user may also reference INSERT or INCLUDE files for portions of the model or for fluid properties. Output files retain the standard SINDA/FLUINT format. The following list provides a summary of all file types.

TABLE 1. File Types

File name/suffix[a]	Description
*.xml.smdl	Main Sinaps model file, XML format
*.tdp	Thermophysical property data file
*.inc	Include (or INSERT) file
*.sin	Text-based SINDA/FLUINT input file, generated by Sinaps
*.pp	Preprocessor out file

TABLE 1. File Types

File name/suffix[a]	Description
astap.for	Compiled fortran listing of the model, generated by SINDA/FLUINT
messages_*.txt	Message file for compiler
messages_*.log	General message file
*.out	Primary SINDA/FLUINT output file
*_csr	Compressed Solution Results (CSR) folder
*.sav	Binary SINDA/FLUINT SAVE file (for restarts, post-processing)
*.usr	Optional user-defined output file
*.ezp2	EZXY® Plot file
*.ezps2	EZXY® Plot Set file
*.subm	Exported Sinaps submodel
*xml	Exported register file, tables, pipe library, or turbomachine library

a. The case name is designated by an asterick (*) in the files names to create unique files for each case.

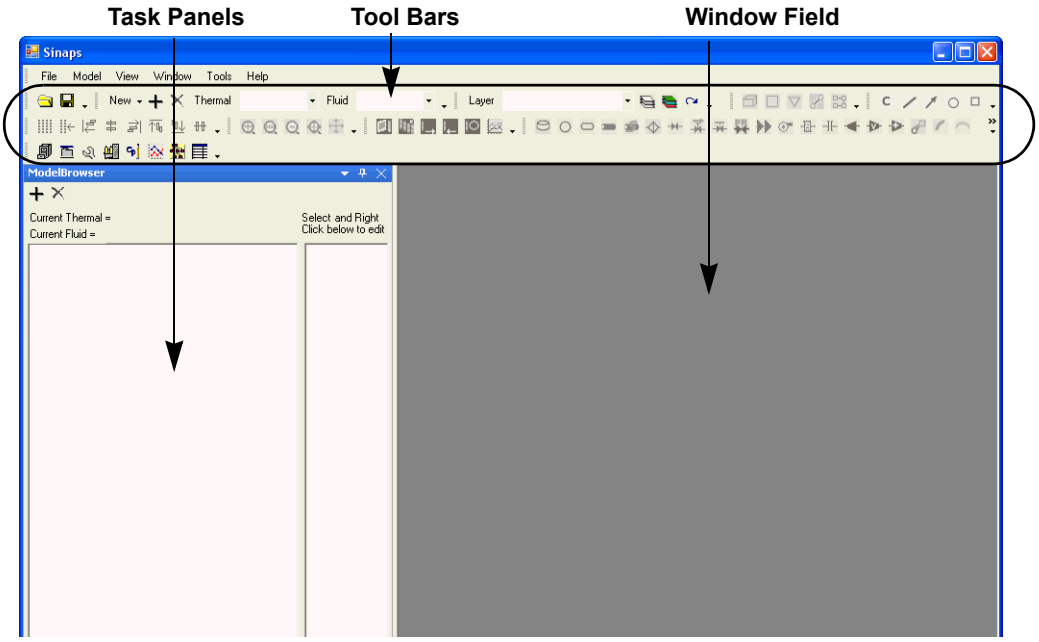
The thermophysical property (*.tdp) data files is designed to be compatible with C&R Thermal Desktop's file format. This allows the user to share a common property data between both applications. However, there are restrictions on the ability to exchange thermophysical property data between the two applications. Sinaps does not support certain Thermal Desktop property features such as anisotropic materials, pressure-dependent properties, effective emissivities, ablation or fusion. When importing such materials from a Thermal Desktop *.tdp file, some data will be dropped:

- For anisotropic materials, only the x-direction properties will be imported.
- Pressure-dependent conductivity will be ignored, resulting in the import only of the value in the conductivity field (the value grayed-out in Thermal Desktop).
- For materials defined using Thermal Desktop symbols or expressions, the symbols and expressions will not be imported. However the *value* of the expression will be imported if possible. Otherwise, the value will be set to unity.
- Effective emissivity is not imported.
- Albative properties are not imported.
- Fusion properties are not imported.

3.2 Overview of the Sinaps Workspace

Sinaps provides and all-in-one integrated workspace comprised of multiple panels and windows in a main application window. The default workspace (shown in Figure 3-1) can be customized by grouping and docking various panels to best suit the user's work habits. Further customization includes setting background colors or image, grid size, behaviors etc. For more information on customizing the workspace see "Customizing the Workspace" on page 3-15.

FIGURE 3-1. **Default Sinaps Workspace**



3.3 Task Panels and Windows

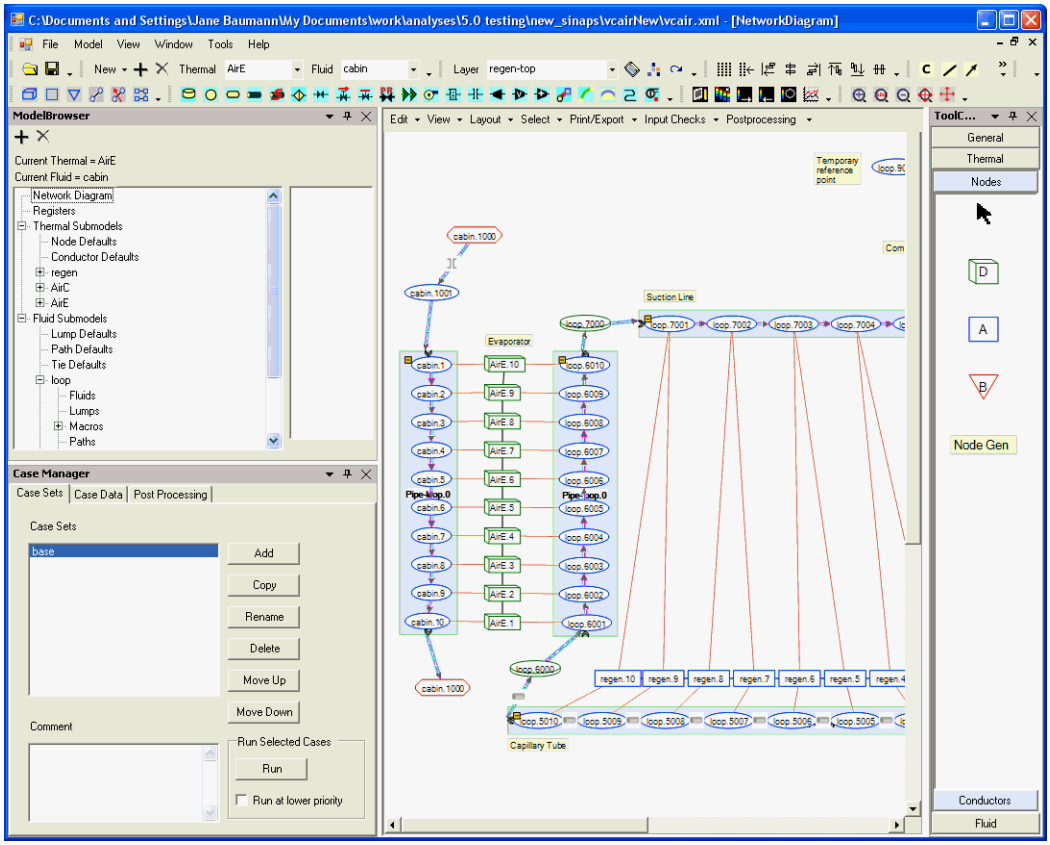
The workspace contains the main menus, tool bars and window field. The window field is where various task panels and windows will open. In the default (initial) layout the visible task panels are the *Model Browser* and the *Case Manager,* which are open on the left side of the workspace. Additional task panels include the (Object) *Properties, Thermophysical Properties* and *Tool Pallet,* all of which are closed in the initial layout. The ***window field*** is the remaining portion of the workspace (shown in gray in Figure 3-1). This field is where various data windows and task panels will open when working in Sinaps. These windows include the *Network Diagram, Registers*, and *logic blocks,* just to name a few. Like many windows-based applications, windows can be minimized, maximized or closed using the icons in the upper right corner of each window. The task panels can also be docked, undocked, relocated outside the workspace, and hidden.

3.3.1 Data Windows

The primary data window is the *Network Diagram*, which occupies the most space in Figure 3-2. This key window is accessible through the model browser by double clicking on Network Diagram, or by using the View>Network option. The ***network diagram*** provides a sketch pad interface for the user-created heat transfer and fluid flow schematic of the model, similar to an electrical schematic. The diagram shows thermal capacitances and fluid mass (nodes and lumps) connected by heat, mass or energy flow connections. The Sinaps network is comprised of *network objects* (nodes, lumps, and comments) which are connected by *links* (conductors, heat transfer ties, flow paths, etc.) The network can be arbitrarily depicted, based on the user's preference and on the detail required for a model. Layers, comments, and collections (collapsible subnetworks) aid in the organization and visual clarity of the network. The user can use icons

along the top frame of the workspace or icons in the tool control task panel shown on the right side of the workspace in Figure 3-2. Details on working with networks can be found in “Working with Networks” on page 4-1.

FIGURE 3-2. Sample Network Diagram (shown with Task Panels hidden)

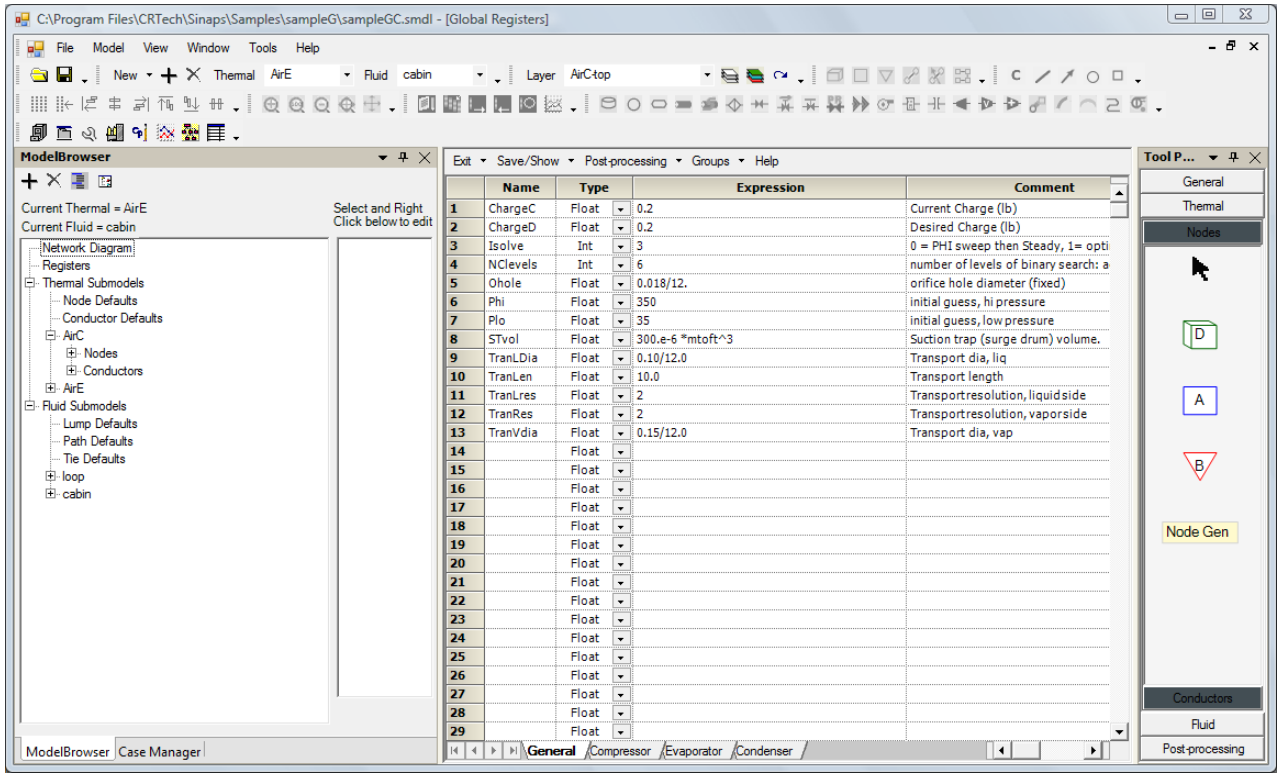


The ***Register Spreadsheet*** (Figure 3-3) is another important data window accessible through the model browser. The register spreadsheet provides a table in which the user can define variables to be used throughout the model in any data entry field. These variables can be interdependent (i.e., functions of each other). They can also be defined as a function of ***processor variables***.* Changing the value of variable such as a dimension or heat load in register data will automatically propagate the change throughout the model. For more information, see “Parameterizing a Model” on page 6-1.

Other windows which will appear in the window field are logic block windows for user-defined logic or subroutine data, array data, control data, etc.

* Processor variables are values such as problem time (TIMEN), the current temperature of a node, etc. that cannot be known until a solution is executing. Refer to the SINDA/FLUINT User’s Manual, Section 2.8 on registers and expressions, for more details.

FIGURE 3-3. Register Spreadsheet

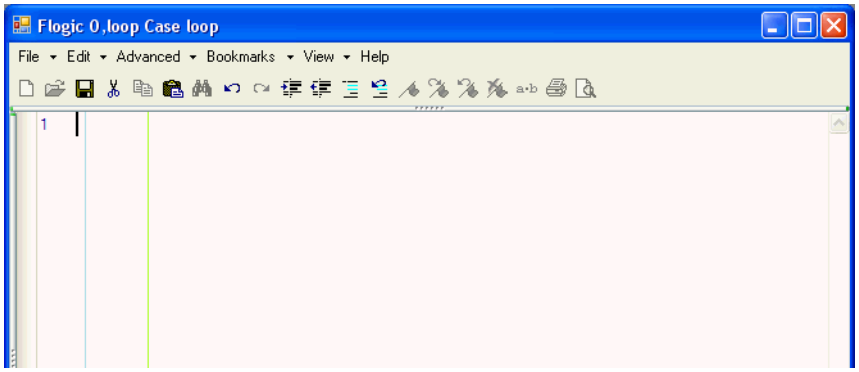


3.3.2 Text Windows

Many of the options in Sinaps will open a text-based input window as shown in Figure 3-4. This is a simple text editor which allows the user to define text-based input such as logic or arrays. The text window is used for all text-based logic blocks such as Operations, Procedure, Output Data, Variables, Flogic, and Subroutines. Logic blocks are used to pass executable instructions to the program in the form of a Fortran-like language. In fact, logic blocks are translated into real Fortran and then compiled and executed. While full details are available in the SINDA/FLUINT manual, for convenience, a summary of rules which apply to these blocks is provided:

1. A “C” in column one designates the line as a comment,
2. A “$” designates any text to the right of the sign to be an inline comment,
3. All text is case insensitive,
4. Fortran column conventions must be obeyed: columns 1-5 are for statement numbers, and column 6 is for continuation marks. *A tab should be used to begin each line not requiring numbering or continuation marks.* Color coding (explained below) is available to help obey these conventions.
5. FSTART and FSTOP commands (starting in column 1) can be used to turn translation on or off. (Caution: turning off translation in this manner also turns off the SINDA/FLUINT spelling checker.)

FIGURE 3-4. Sample Text Window



The text editor is enhanced with color coding to improve the readability and context of the text. The blue vertical line (Figure 3-4) indicates column one, while the green line indicates column 7. Syntax highlighting (Table 2) improves readability of the content and helps reduce errors by visually depicting code structure.

TABLE 2. Automatic Syntax Highlighting

Content	Color
comments (C in column 1)	green
fortran built-in functions	red
fortran keywords	blue
logical operators	brown
strings (in single quotes)	red

The text editor provides several useful features beyond basic copy and paste functions, such as:

- Bookmarks - bookmarks are useful for flagging certain lines in logic. Once placed, the user can move to the next or previous bookmark to quickly move through a large file.
- Comment a region (under Advanced) - the user can select a block of code and comment or uncomment all the lines.
- View White Space (under Advanced) - this toggle option will display all blank spaces as dots and all tabs as arrows.
- Make uppercase/lowercase (under Advanced) - these options will convert the text in a selected region to all upper or all lower case.
- Tabify/Untabify Selection (under Advanced) - these options will convert white spaces to tabs if appropriate or convert tabs to spaces.
- Increase/decrease Indent (under Advanced) - this option increases or decrease indentation for a selected region of text.
- Tabs - User-defined tab location (can be changed using File>Options>Behaviour>Tabs)

The Options dialog (File>Options) allows the user to customize the interface such as adjusting tabs, toggling line numbers, toggling column guides, etc.

3.3.3 Task Panels

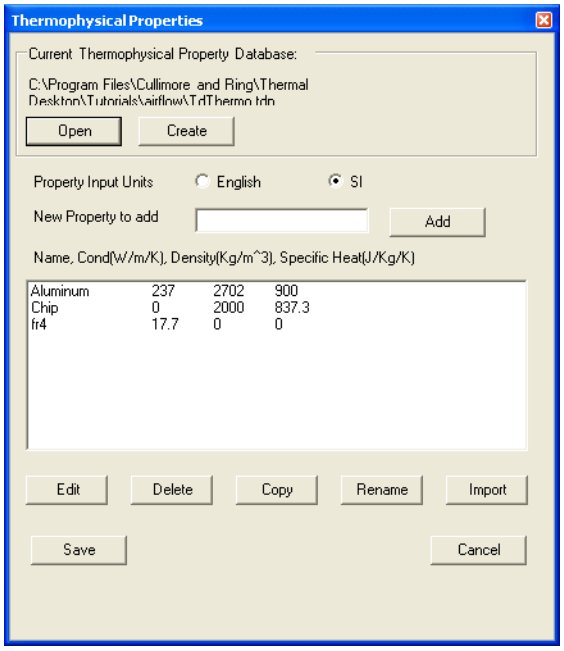
Task Panels are windows in the workspace space which can be docked, undocked, relocated, hidden, or closed by the user. Task Panels are used to display the Model Browser, Case Manager, Thermophysical Properties, Object Properties and Tool Pallet.

The ***Model Browser*** shown in the upper left portion of Figure 3-3, provides a tree-type structure of the entire model arranged by submodel and object type. It also allows access to the network diagram and the register window. The model browser provides a means of navigating the entire model, finding/editing objects, setting defaults, adding/editing logic and arrays, and defining working fluids for fluid submodels. Full documentation of the model browser can be found in "The Model Browser" on page 8-1.

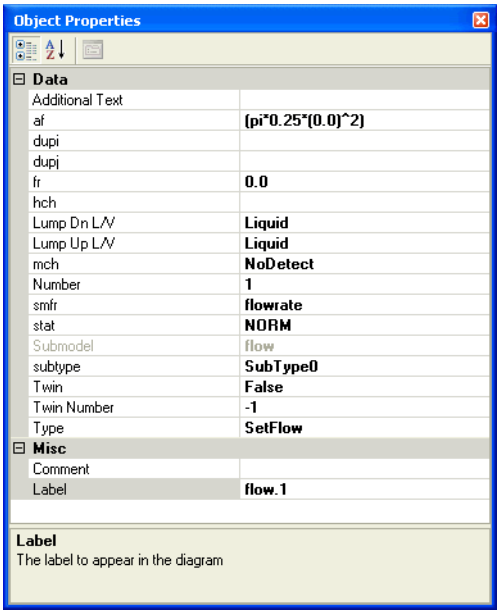
The ***Case Manager***, shown in the lower left portion of Figure 3-3, allows the user to set up, control, launch, and customize multiple runs or "cases" for a single model. The user can specify model units and output files, set up steady state, transient and parametric runs, set control values, vary the model configuration by case, control post-processing, and create user-defined logic and subroutines. Full documentation of the case manager can be found in "Using the Case Manager" on page 9-1.

The ***Thermophysical Properties*** window, shown in Figure 3-5, is accessible through the pull down menu `View>Thermophysical Properties`. This option allows the user to create a thermophysical property database comprised of conductivity, specific heat, and density for materials that will be needed in the model. By default, an empty property file (named "TdThermo.tdp") is created in the working directory when a model is first created. The user has the option of either opening a new or existing thermophysical property file, or adding material properties in the default database file. Using an existing database is useful if several analysts are sharing a common property database, with some of those analysts perhaps using C&R Thermal Desktop® (since the *.tdp files are formatted the same in both tools). For more information on defining thermophysical properties, see "Using the Case Manager" on page 9-1.

FIGURE 3-5. Thermophysical Properties



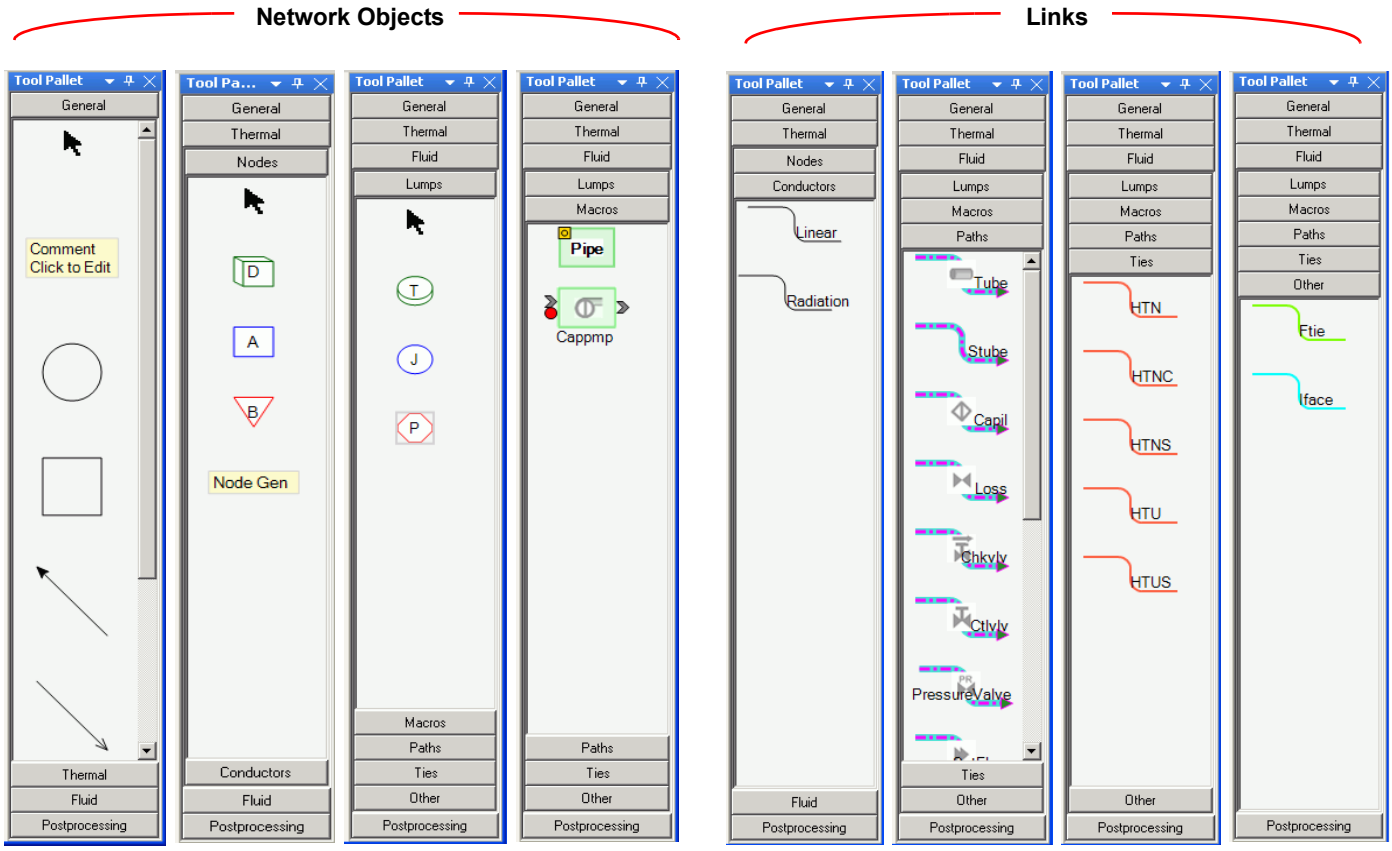
The ***Object Properties*** task panel allows the user to edit any element of the network including icons (e.g., lumps, nodes, collections), links (e.g., conductors and flow paths), and the network window itself. Using this task panel, the user can edit any network element as an alternative to the more specialized edit forms available for most network elements. This option provides advanced capabilities such as editing the appearance an object (e.g., the image used for the icon), the color and image of a link between two objects, and much more. Using Object Properties, the user can customize the network window by changing network display characteristics, such as adding a background image.



Note: In general, the Object Properties dialog is considered an advanced feature within Sinaps. However, some network characteristics and object properties can only be accessed using the Object Properties dialog. Examples of items that only accessible using the Object Properties dialog include network background, post-processing color bars, and

The ***Tool Pallet***, shown on the right side of Figure 3-2, is shown expanded into a variety of modes in Figure 3-6. This panel provides selectable icons for building network diagrams. It contains an accordion of several expandable buttons that organize the various types of network objects and links. The user can select objects, which will then be the default object placed on the network diagram in subsequent pick operations. Alternatively, the user can drag network objects (comments, nodes, lumps, or macros) to the drawing field. As an alternative to the tool control, the user can use the icon-based tool bars described in “Toolbars” on page 3-13.

FIGURE 3-6. Tool Pallet Task Panel



3.4 Menus and Toolbars

Sinaps provides pull-down and pop-up style menus depending on which task panel or window is active.

Pull-down Menus: Pull-down menus are selected from the menu bar that appears near the top of most windows. The select button is used to choose the desired option, but this button need not remain pressed while navigating the sub-menus that appear. (The options displayed on a menu differ for each window, with the important options documented in the manual sections that describe those windows.)

To use a pull-down menu, move the cursor over the desired option on the menu bar, and either

1. Click the *select* mouse button once, then move the cursor to highlight the desired selection and click the mouse button once more. Or, if more submenus appear, click the mouse when the cursor is over the desired suboption.
2. As an alternative to the above method, the *select* mouse button can be pressed down and held, and released when the cursor is over the desired option or suboption.
3. A hybrid of the above two methods can be used for selecting submenus, which will then remain on the screen while the options underneath that submenu are navigated using either of the above methods.

Pop-up Menus: Pop-up menus are generally accessed by pressing the *operate* (right) or middle mouse buttons. The options that are presented will be dependent upon the cursor location at the time the mouse buttons are pressed.

3.4.1 Sinaps Main Menu Options

The primary Sinaps workspace window includes a set of pull-down menus along the top frame of the workspace (File, Model, View, Window, Tools, Help). These menus provide access to various task panels and windows and provide the ability to set preferences.

The Model pull-down provides access for starting new models, adding (+), deleting (X), renaming, exporting and importing of submodels, setting model defaults, accessing the notes file, and exporting a text-based file of the model (a SINDA/FLUINT input file).* The Model>Notes option opens an text edit window useful for model documentation, revision history, etc. See

The View menu provides access the Model Browser, Object Properties, Tool Pallet, Case Manager and Thermophysical Properties, along with the EZXY® Plotter, the Network Diagram and the Global Registers window.

The Window menu provides a variety of options for organizing the windows in the Window Field. When windows become numerous or overlapping, the Window menu permits viewing of all open windows through tiling or cascading. It also provides a means to locate and bring the any particular window forward when several windows are open. The Tabbed option organizes all windows in a tabbed fashion versus the default floating windows. When tabbed, one can easily move between open windows by clicking on the tabbed option. This is similar to the tabbed option available in

* This option is intended to be a quick short-cut. It generates the currently selected case if more than one exist, and if only one has been selected. (If no cases are selected, or if more than one is selected, then this option generates the model corresponding to the first case in the Case Manager, ignoring any selections). Any specific case can be exported to a SINDA/FLUINT input file using the Write File button under Case Data>General.

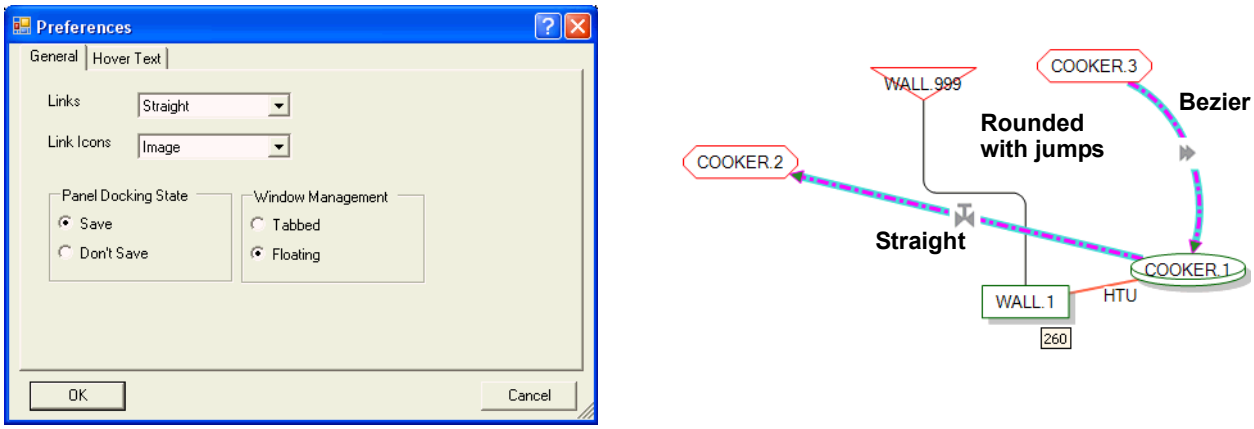
preferences but is not retained between Sinaps sessions. For more information see “Window Management” on page 3-17.

The Tools option provides access to setting user preferences, and find/replace tools. It also provides a location to read an existing SINDA/FLUINT input file into Sinaps. These options are further described in the subsections below.

3.4.1.1 Setting Preferences

The user has the option to set preferences (Figure 3-7) for controlling the physical appearance of links in the network, window docking states, window management, and hover text choices (Figure 3-8). ***(Hover text*** will appear when the user places the cursor over an object in the network diagram.) Preference settings are stored in the registry so they are retained when Sinaps is reinstalled or updated.

FIGURE 3-7. Setting User Preferences



Network ***links*** are the lines that connect various objects in the network diagram. These represent conductors in a thermal submodel and flow paths, convection ties, fluid-to-fluid ties (FTIEs), or control volume interfaces (IFACEs) in a fluid submodel. Links can take the form of straight lines, bezier (curved) lines, or rounded lines with jumps. Links can optionally be labeled using either text or icons. The preferences option defines the shape for any subsequently generated links; once generated, the link shape can be changed by highlighting the link and using the pop-up (right button) menu.

The user has the option of saving the docking state of windows in a model. By default, the docking states are not saved. If the user selects the save option, the docking states (i.e., whether it is docked or not, and where) will be saved with the model and the next time the model is opened the task panels will be docked as they were when the user last saved the model.

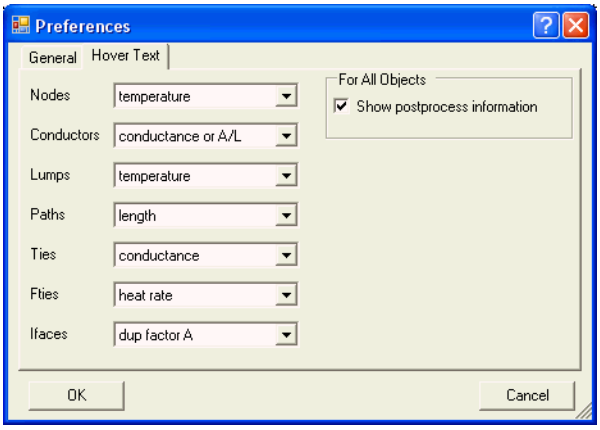
Window Management allows the user to toggle between floating and tabbed windows in the window field. When tabbed, the windows will be tabbed across the top of the window field and the user can easily switch between windows. (Under the Window pull-down menu, the user can temporarily change the windows to be tabbed without changing how this model will appear next time it is opened.) For more information see “Window Management” on page 3-17.

3.4.1.2 Hover Text

The network diagram has the option of using hover text on all network objects (i.e., lumps, nodes, conductors). ***Hover text*** is the small text box that will appear on the screen when the mouse cursor is held over any object. Using the preferences form or the network menu option Input Checks>Set Parameters, the user may select the type of data (temperatures, capacitance, conductance, pressure, quality, flow rate, etc.) the hover text will display. If the parameter to be displayed is an input value that is defined via an expression, the hover text will display the expression in lieu of a value.

The user has the option of applying the hover text controls to post-processing features (see "Output and Post-processing" on page 10-1). If the Show Post-process Information box is checked *and* the network is in a post-processed view, the hover text will be controlled by the post-processing options defined in the case manager ("Submodel-specific Control Data" on page 9-31) *instead of* the hover text options selected in the Preferences window. Note that only parameters set for *color* post-processing will appear as hover text. Hover text does not recognized thickened post-processing options.

FIGURE 3-8. Setting Hover Text



3.4.1.3 Read in a SINDA/FLUINT Input File

This option provides a means of importing an existing SINDA/FLUINT file, rather than creating a model from scratch using the Model>New option. Details on this option are provided in "Importing SINDA/FLUINT Text Files" on page 11-3.

3.4.1.4 Find and Replace

The Tools>Find and Replace option is useful when changes have been made to the names of logic variables and especially to registers, which can be present in text blocks (logic and array data), and in input fields (perhaps within an expression). For example, if a register were named "bobsled," but were renamed "rosebud," then this option could be used to replace any occurrence of "bobsled" with "rosebud" in expressions, text blocks, etc.

This option is global in scope. Therefore, the Match Whole Word suboption is on by default to protect against accidental renaming. For example, without that selection, changing "is" to "is2" would also change "istoo" to "is2too" globally. Additional control, in the form of matching case, is available.

For more aggressive control, at the risk of accidental renamings, regular expressions (such as wild cards) can be used. For example, replacing “Hot*” with “Hotter*” will change “Hotest” to “Hotterest” but will also change “Hot1” to “Hotter1.”

3.4.2 Toolbars

Toolbars are located across the top of the main Sinaps workspace window under the pull-down menus. Each icon in a toolbar has associated hover text which will identify the purpose of the icon. The main toolbars are summarized in Table 3 as they appear in the default/initial set-up, though the user can perform extensive customizations (“Customizing Toolbars” on page 3-20).

TABLE 3. Standard Toolbars

Toolbar Name	Actions
File	Open
	Save
Model	Create new model
	Add a submodel
	Delete a submodel
	Current thermal submodel
	Current fluid submodel
View	Open/view Model Browser
	Open/view Object Properties
	Open/view network object Tool Pallet
	Open/view Case Manager
	Open/view Thermophysical Properties
	Open external plotter (EZXY)
	Open/view Registers
Layers	Current layer pull down
	Edit layers
	Color by layer
	Reset colors

Network Toolbars: In the default layout, there is a series of toolbars which only appear when the network diagram is open. These toolbars are summarized in Table 4.

TABLE 4. Network Toolbars

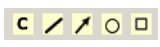
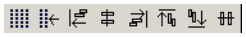
Toolbar Name	Actions
Thermal	Create a diffusion node
	Create an arithmetic node
	Create a boundary/heater node
	Create a linear conductor
	Create a radiation conductor
	Generate nodes and conductors
Fluid	Create a tank
	Create a junction
	Create a plenum
	Create a time-dependent flow path (TUBE)
	Create a time-independent tube (STUBE)
	Create a filter or wick (CAPIL)
Fluid (continued)	Create a loss (LOSS)
	Create a directional dependent loss (LOSS2)
	Create a check valve (CHKVLV)
	Create a control valve (CTRVLV)
	Create a downstream pressure regulating valve (DPRVLV)
	Created an upstream pressure regulating valve (UPRVLV)
Fluid (continued)	Create a set mass flow device (MFRSET)
	Create a set volumetric flow device (VRFSET)
	Create a pump (PUMP)
	Create a tabular device (TABULAR)
	Create an orifice ()
	Create a turbine (TURBINE)
	Create a compressor (COMPRESS)
	Create a positive displacement compressor (COMPPD)
Fluid (continued)	Create a heat transfer tie (TIE)
	Create a fluid-to-fluid heat transfer tie (FTIE)
	Create a control volume interface (IFACE)
	Create a Sinaps Pipe
	Create a capillary evaporator pump (CAPPMP)
Drawing 	Create a comment
	Draw a line
	Draw an arrow
	Draw a circle
	Draw a rectangle

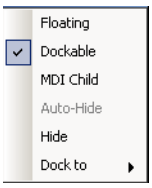
TABLE 4. Network Toolbars

Toolbar Name	Actions
Zoom	Over overview window
	Zoom in
	Zoom out
	Zoom 100%
	Zoom Extents (fit network in window)
Grid and Alignment	Show grid
	Snap objects to grid
	Align left sides
	Align centers vertically
	Align right sides
	Align top sides
	Align bottom sides
	Align centers horizontally
Post-processing	Open Post-processing Manager
	Post-process selection set (default all)
	Show next time step
	Show previous time step
	Animate through time
	Make an XY plot from selections

3.5 Customizing the Workspace

3.5.1 Docking Task Panels

The task panels, unlike data windows, are dockable along the perimeter of the data field in the Sinaps workspace. Panels can alternatively be combined to create *tabbed* panels within the workspace. Dockable panels are easily identified by right clicking on the top title bar of the panel. If the panel is dockable, the menu shown below will appear.



Three types of window formats are available task panels, Floating, Dockable, and MDI Child. A ***Floating*** panel can be placed anywhere on the screen, including outside of the Sinaps application window. In the floating format, a panel is not dockable but can be hidden. ***Dockable*** panels can be docked to the frame of the application window or tabbed to

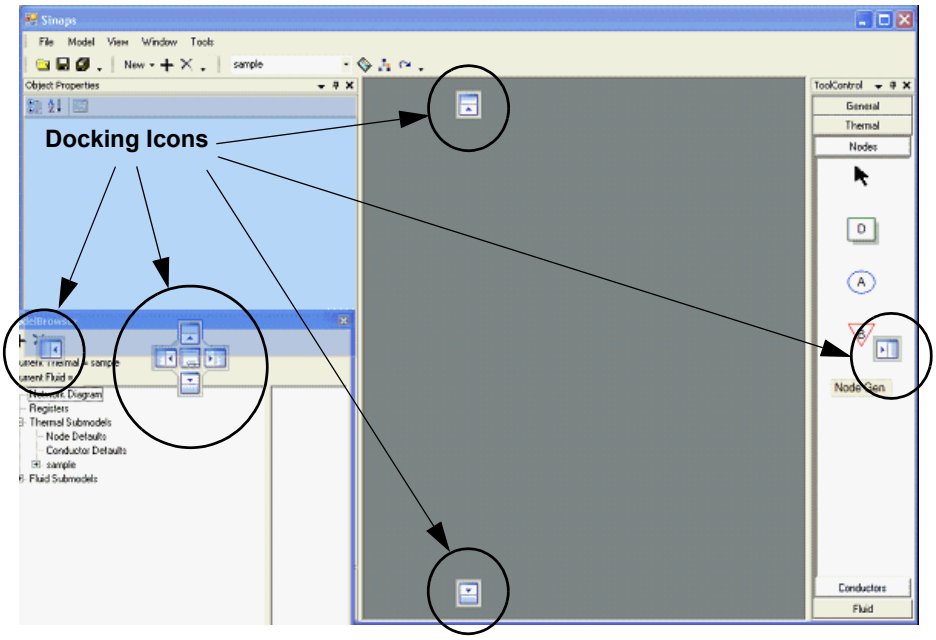
other dockable panels. When undocked, dockable panels behave similar to floating panels. When docked, dockable panels can be autohidden. The ***MDI Child*** option created a child window to the parent application. As a child window, the task panel is located within the parent application window and cannot be moved outside of the application. In the MDI Child format, a task panel cannot be docked but can be hidden.

To undock a panel, simply right click on the title bar of the panel to access the docking menu and then select Floating. The panel will change to a floating window and can be moved around the screen using the cursor. Alternatively a docked window can be undocked simply by left clicking on the title bar and dragging it into the data field.

> Note: Using the docking menu to undock a panel changes the status of the panel from "Dockable" to "Floating" while dragging a panel off a docking station does not change the status. If a panel has been changed to "Floating," it cannot be redocked unless the menu toggle is changed to "Dockable."

To redock a floating or and MDI Child window, use the docking menu: click on the title bar to set it to Dockable. A "dockable" panel can be docked by selecting Dock to and then the location where the panel is to be docked (Dock to>[location]). Alternatively a dockable panel can be dragged to the area to be docked. As the panel is dragged across the data field, docking icons will appear and a portion of the window turn blue (Figure 3-9). The blue shaded area provides a preview of where the panel will dock. In general, the user can dock on the sides of the data field or within any of the other task panels. In addition, any panel can become a tabbed option within any other panel.

FIGURE 3-9. Docking Task Panels



3.5.2 Hiding Task Panels

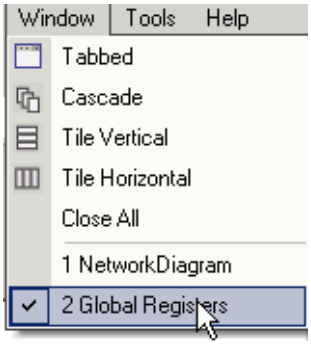
Once a task panel has been docked to the frame of the workspace, it can be hidden or collapsed using the push pin icon in the upper right corner of the panel. After clicking the icon, the panel will "auto-hide" and convert to an expandable tab along the frame at the docking point. Alternatively, the panel can be collapse using the Auto-Hide option on the docking menu. To view the panel after collapsing it to a tab, simply position the cursor over the tab and the panel will expand. When you move your cursor off the panel to a different portion of the workspace the panel will automatically returned to the tabbed or hidden state. When a hidden panel is expanded, you can click the push pin icon again to keep it open in the work space.

The Auto-Hide and Hide features on the docking menu are different. The Auto-Hide collapses the panel to a tab along the frame of the data field. The Hide option closes the panel. To reopen a hidden panel, use the pull down menu option View>[panel name].

3.5.3 Window Management

The user can choose a preferred type of window manager through the preferences dialog (see "Setting Preferences" on page 3-11). By default, windows such as the network, registers, and logic blocks will appear as floating windows in the window field as shown in Figure 3-10. These windows can be minimized, maximized, tiled, or cascaded.

An alternative window management option is ***tabbed windows***. When working with tabbed windows, the cascade and tile options in the window menu will have no effect. When using this option, all windows in the window field will be maximized and tabbed along the top frame as shown in Figure 3-11. Tabbed windows can be selectively grouped as shown in Figure 3-12 where for example the network, operations, and design data are tabbed together in the top portion of the window and registers is tabbed separately in the lower portion of the window. Both horizontal and vertical tabbed groups can be created. To create tabbed group, right click on the upper tab for a window and select either New Horizontal Tabbed Group or New Vertical Tabbed Group. The move option allows you to shift any window to a different tabbed group.



The General tab on Tools->Preferences sets windows management options that are saved along with the model, and that will apply by default the next time the model is opened. Options under the pull-down Windows menu are temporary (i.e., not saved along with the model), and are intended to be used to help navigate through a model. An example of these navigation and arrangement options is shown at the right. The top part of the menu offers convenient arrangement options as discussed above. The bottom of the Windows menu allows the user to bring a specific window to the top (Global Registers is being selected in the example shown at the right). Windows that are open (but potentially obscured) are listed at the bottom of this form; this form can bring windows to the top, but is not used to open them in the first place. The user is urged to take advantage of these navigation features, and to avoid closing windows in order to clean up or organize the Sinaps environment since it might take several steps to find and reopen a window. In other words, leave windows open until it is clear that they are no longer needed.

FIGURE 3-10. Floating Window Management

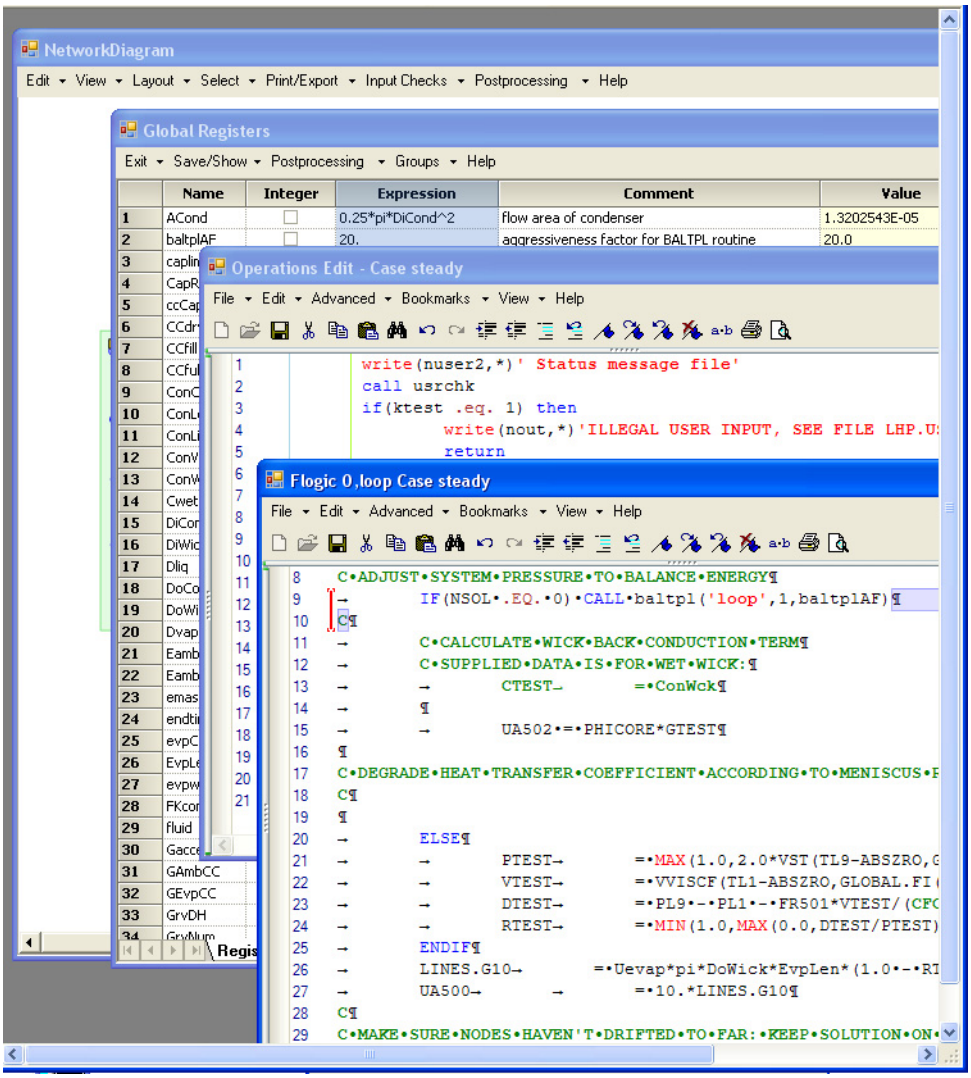
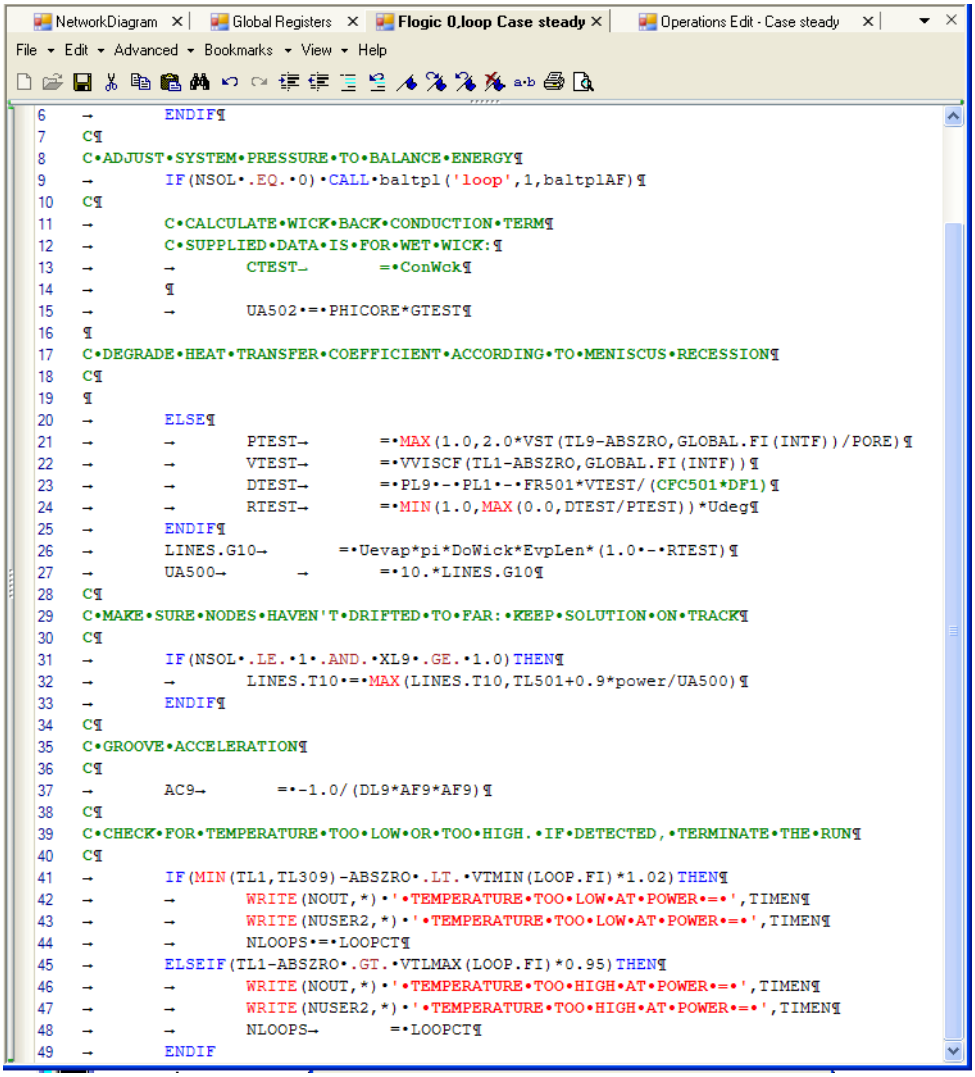
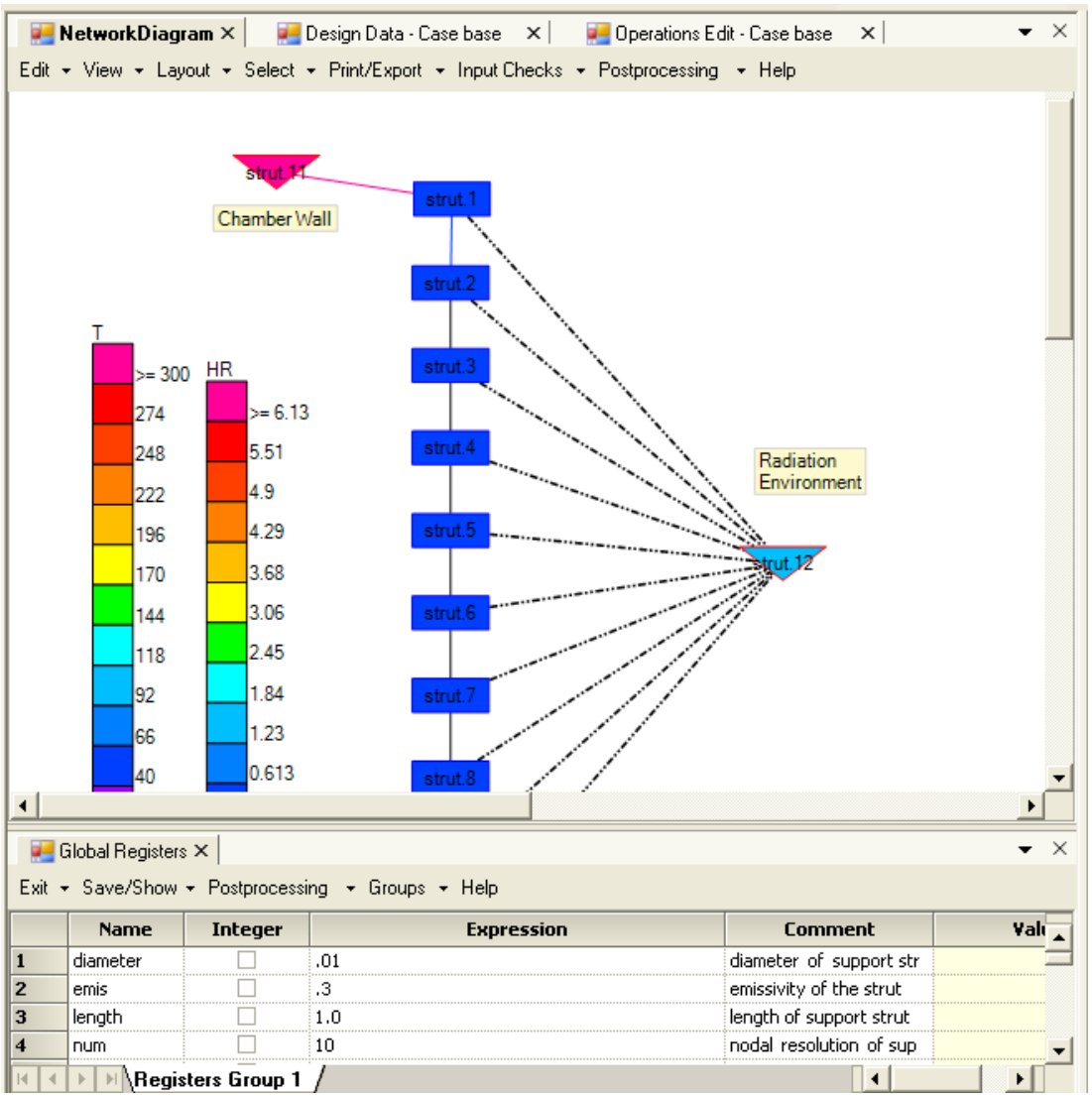


FIGURE 3-11. Tabbed Window Management




```
6   →       ENDIF¶
7   C¶
8   C•ADJUST•SYSTEM•PRESSURE•TO•BALANCE•ENERGY¶
9   →       IF(NSOL•.EQ.•0)•CALL•baltpl('loop',1,baltplAF)¶
10  C¶
11  →       C•CALCULATE•WICK•BACK•CONDUCTION•TERM¶
12  →       C•SUPPLIED•DATA•IS•FOR•WET•WICK:¶
13  →       →       CTEST→          =•ConWck¶
14  →       ¶
15  →       →       UA502•=•PHICORE*GTEST¶
16  ¶
17  C•DEGRADE•HEAT•TRANSFER•COEFFICIENT•ACCORDING•TO•MENISCUS•RECESSION¶
18  C¶
19  ¶
20  →       ELSE¶
21  →       →       PTEST→          =•MAX(1.0,2.0*VST(TL9-ABSZRO,GLOBAL.FI(INTF))/PORE)¶
22  →       →       VTEST→          =•VVISCF(TL1-ABSZRO,GLOBAL.FI(INTF))¶
23  →       →       DTEST→          =•PL9•-•PL1•-•FR501*VTEST/(CFC501*DF1)¶
24  →       →       RTEST→          =•MIN(1.0,MAX(0.0,DTEST/PTEST))*Udeg¶
25  →       ENDIF¶
26  →       LINES.G10→          =•Uevap*pi*DoWick*EvpLen*(1.0•-•RTEST)¶
27  →       UA500→          →       =•10.*LINES.G10¶
28  C¶
29  C•MAKE•SURE•NODES•HAVEN'T•DRIFTED•TO•FAR:•KEEP•SOLUTION•ON•TRACK¶
30  C¶
31  →       IF(NSOL•.LE.•1•.AND.•XL9•.GE.•1.0)THEN¶
32  →       →       LINES.T10•=•MAX(LINES.T10,TL501+0.9*power/UA500)¶
33  →       ENDIF¶
34  C¶
35  C•GROOVE•ACCELERATION¶
36  C¶
37  →       AC9→        =•-1.0/(DL9*AF9*AF9)¶
38  C¶
39  C•CHECK•FOR•TEMPERATURE•TOO•LOW•OR•TOO•HIGH.•IF•DETECTED,•TERMINATE•THE•RUN¶
40  C¶
41  →       IF(MIN(TL1,TL309)-ABSZRO•.LT.•VTMIN(LOOP.FI)*1.02)THEN¶
42  →       →       WRITE(NOUT,*)•'•TEMPERATURE•TOO•LOW•AT•POWER•=•',TIMEN¶
43  →       →       WRITE(NUSER2,*)•'•TEMPERATURE•TOO•LOW•AT•POWER•=•',TIMEN¶
44  →       →       NLOOPS•=•LOOPCT¶
45  →       ELSEIF(TL1-ABSZRO•.GT.•VTLMAX(LOOP.FI)*0.95)THEN¶
46  →       →       WRITE(NOUT,*)•'•TEMPERATURE•TOO•HIGH•AT•POWER•=•',TIMEN¶
47  →       →       WRITE(NUSER2,*)•'•TEMPERATURE•TOO•HIGH•AT•POWER•=•',TIMEN¶
48  →       →       NLOOPS→         =•LOOPCT¶
49  →       ENDIF
```

FIGURE 3-12. Example of Horizontal Tabbed Window Groups



3.5.4 Customizing Toolbars

Like most Windows-based software, toolbar visibility, location, and contents can be modified by the user.

To move a toolbar, place the cursor over the toolbar *background* (not over a button!) and press the left mouse button. While holding down the mouse button, drag the toolbar to the new location.

To customize the contents of the toolbar or to create a new toolbar, place the cursor anywhere in the empty portion of the top frame of the window and click the right mouse button to access the toolbar pop-up menu. Selecting any toolbar name in the menu will toggle the associated toolbar on or off. Selecting Customize will allow the user to customize an

existing toolbar (add or delete items) or to create their own toolbar. Hovering the cursor over any existing toolbar and clicking the right mouse button will directly open the customization dialog.

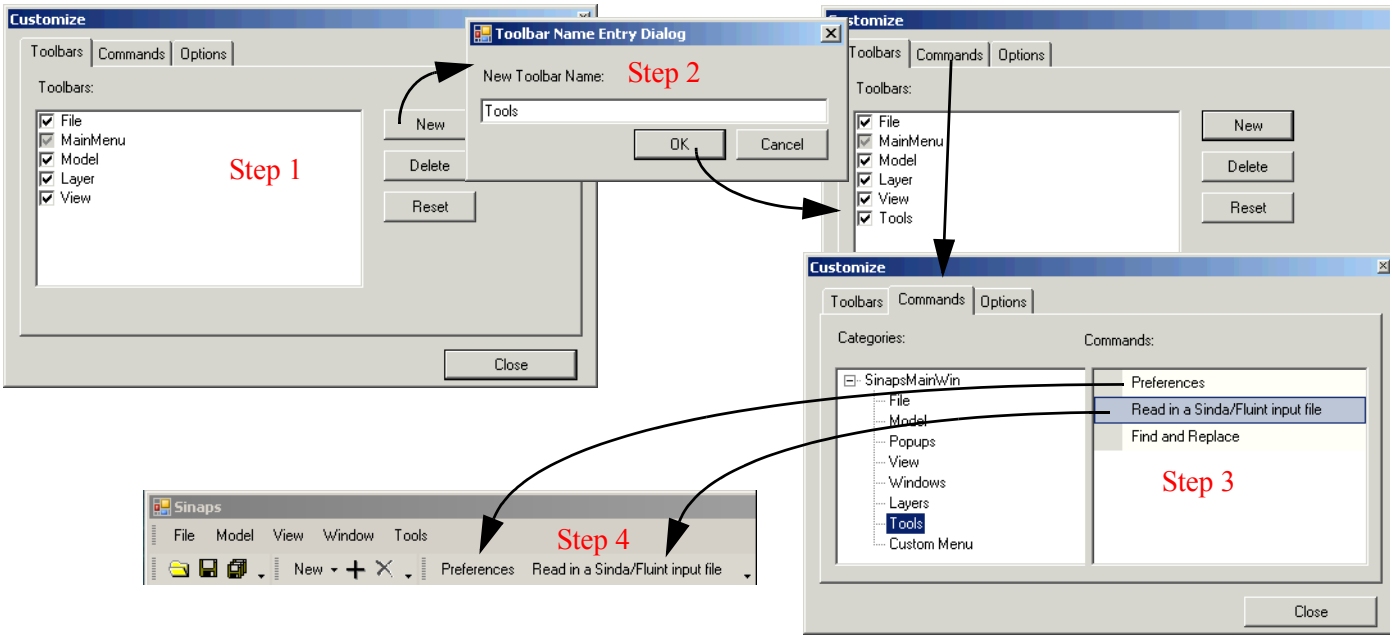
> Note: When the network is closed, some toolbars are grayed out and inactive. These toolbars are network specific and **cannot** be modified by the user. In this case any new "network" icons shound be added to a "new" user defined toolbar. Other toolbars such as Layer and File which are always active, can be modified by the user.

Example: Creating a new custom toolbar for accessing the TOOLS menu options:

1. Place the mouse cursor in the upper frame of the window where existing toolbars are located and click the right mouse button. Select Customize to bring up the toolbar customize window as shown in the upper left of Figure 3.5.4.
2. Select New to bring up the Toolbar Name Entry Dialog. Type in a name for the new toolbar. Click OK. A place holder will be created in the on the frame of the main window to the right of the other toolbars.
3. Select the tab titled Commands to get a list of available commands. The categories listed in the left pane represent the various menus available in Sinaps. Select the menu item (in this case Tools) containing the items to be added to the toolbar. The available commands will appear in the right pane.
4. Using the mouse, drag the various commands to the new toolbar placeholder. The new toolbar will appear as shown in Step 4 of Figure 3.5.4. For this example we have added *Preferences* and *Read in a SINDA/FLUINT input file* to the toolbar. If icons are associated with the command, the icon will appear in the toolbar instead of the text.

If the customization dialog is open, further customization options for a toolbar can be accessed by right clicking any icon. The subsequent menu allows you to change image, show text with the icon, show text only, show image only, change the name of the icon, and create break lines between icons (begin groups). Note that these options apply to both the toolbar icons and pull down menu options for the item being customized. This same customization dialog appears by right clicking on any pull down menu item when the customization dialog is open.

FIGURE 3-13. Creating Custom Toolbars



CHAPTER 4

Working with Networks

This chapter will cover how to create and modify networks, and how to work within the network window. Since model networks can become very complex and difficult to arrange on a single screen view, Sinaps provides many options for viewing and navigating diagrams, as well as various means of managing network diagrams themselves (such as layers, collections, and clones). Sinaps also provides several layout editing options for organizing the network such as alignment tools, a variable grid, zoom features, and automatic layouts.

4.1 Working with a Network

The network window is accessible from the Model Browser or from the View>Network menu option. The default set of network element toolbars will appear when the network window is opened.

This section will briefly introduce concepts used for creating thermal/fluid networks; detailed information on the various network objects can be found in “Creating Thermal/fluid Networks” on page 4-5. To create a thermal/fluid network, the user may use either the toolbar icons or the Tool Pallet. The Tool Pallet panel, which is especially useful during initial model generation, is accessible from the pull down menu View>Tool Pallet. The Toll Pallet organizes the network elements by type using an expandable accordion bar (Figure 3-6).

Modeling elements are comprised of ***objects*** (nodes and lumps) which can be interconnected using a variety of ***links*** (conductors, paths, ties, etc.). For thermal models, the user creates a conductance-capacitance schematic of the model using node objects and linking them together with conductors (heat flow paths). Analogous (but more complicated) objects and links are used to represent concurrently solved fluid flow networks.

Once placed on the network diagram, objects such as nodes can be moved, edited, or deleted. If an object is moved, all associated links to other objects are stretched/shrunk accordingly. If an object is deleted, all links between it and other objects are also deleted.

Hover test appears when the cursor is placed over any object or link on the network. The hover text can be set to display a variety of parameters associated with object or link based on settings defined by the user in hover text properties, see “Hover Text” on page 3-12.

> Note: The Network Diagram window has its own View pull-down menu at the top, which is not to be confused with the View option at the top of the entire Sinaps workspace. The workspace View option applies to all of Sinaps, whereas the Network Diagram View option applies only to the network.

4.1.1 Creating Comments and Drawing Objects

To help organize the network, Sinaps provides a variety of tools such as layers, collections, comments, and drawing tools. Comments and “drawing objects” are found on the General panel of the Tool Pallet.

Comments may be place on the diagram at the user’s discretion to provide object identification or applicable notes. Once a comment is placed on the drawing field, clicking on the comment frame will select the comment for moving, deleting, etc. Clicking in the comment text area will activate the comment text editor.

Drawing objects (arrows, lines, boxes, circles) are visual comments: they are not related to modeling objects since their presence or absence does not affect model results. They can be used to explain or document a diagram. Drawing objects can be resized, moved, and deleted.

Drawing objects can be edited through the object properties dialog to change the color and pen width. Line and arrows, can also be edited for their end point appearance. To edit a drawing object, open the object properties dialog (main menu View>Properties) and then select the object in the network diagram. The object properties panel will be refreshed with information on the object to be edited.

> Tip: Any time the cursor is clicked in the open space of a network diagram, the last object selected in the Tool Pallet or a toolbar will be generated. This functionality is intended as an aid in rapid model building, but it can lead to the generation of unwanted objects and links. To disable this feature, choose the selection arrow at the top of the General, Node, or Lump panels of the Tool Pallet.

4.2 Creating and Managing Layers

Layers allow the user to turn on and off the visibility of portions of the network. The network objects may be assigned to any number of named superimposable layers, and the visibility of each layer may be toggled on and off as needed, whether on the screen or as printed. When a layer is *inactive*, not only are its members invisible, but other objects may occupy that now-empty space when using various automatic layout tools.

Layers may be collected into ***layer sets*** for even higher level manipulations.

When objects are dropped into the network, they will be placed in the ***current layer***. The designation of which layer is current can be changed as needed, as will be described below (Section 4.2.2.1).

Layers can be created or deleted, and objects can be moved between layers. Unlike network objects such as nodes or lumps, network links (such as paths and conductors) are not associated with any particular layer. Thus, if two objects are on distinct layers, a link connecting those two objects will only be displayed if both layers are active.

4.2.1 Automatic Layer Generation

Sinaps automatically creates layers in the network diagram based on submodels to assist the user in managing the overall network. When a submodel is created, an associated layer of the same name is created and set to be the current layer. Thus, when objects are dropped into the network, they will be placed in the current layer which is the layer associated with the most recently created submodel.

When reading in a SINDA/FLUINT text based input file or importing a model from Sinaps*Plus*, a separate layer will be created for each thermal and fluid submodel, LINE macro, and HX macro.

4.2.2 Creating and Managing Layers

The Layer Edit menu is accessible through the pull down menu in the network window (View>Edit Layer) or by clicking on the Edit Layer icon in the layer tool bar (.). The Layer Edit window or ***Layer Manager***, shown on the left side of Figure 4-1, is a tabbed form providing access to *layer* management and to *layer set* management (the latter of which will be described below ... see “Layer Sets” on page 4-4).

Using the Layers tab, the user can add new layers, designate a layer to be the *current layer*, add, rename, delete or merge layers, toggle layers on or off, and set them visible or invisible for printing using the check boxes.

To create a new layer, click New and provide a name for the new layer in the New Layer Name dialog. The new layer will automatically become the current layer in the network diagram. When the Layer Edit window is later reopened, the layers will have been alphabetized by name.

To delete or rename a layer, highlight it in the list then press Delete or Rename, respectively. (The current layer cannot be deleted.)

To merge a layer with the current layer, highlight it in the list then press Merge. A prompt will appear providing the option of leaving the old layer empty (rather than deleting it) after all objects have been moved from it into the current layer.

The check boxes in the On and Print columns of the layer manager set the visibility of each layer for network viewing and printing respectively. The visibility of the current layer must always be On. To turn off the visibility of the current layer, a new layer must first be assigned as current (Section 4.2.2.1).

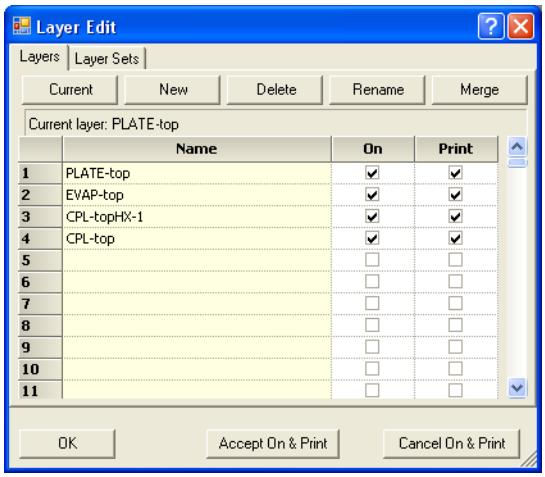
When a layer is “off” it is in a frozen state. Any changes to the network, such as selecting everything and then moving the icons, will not apply to objects on layers which are turned off. The exception to this is post-processing. If the entire network is post-processed while some layers are turned off, these layers will be post-processed upon being reactivated.

In the Layer Manager, *clicking* OK *button will accept all changes and exit the form.* Accept On & Print accepts the changes, applies the changes to the diagram for visual verification but keeps the form open to allow further changes to be made. Cancel On & Print will exit the form without applying the changes.

Any merge, delete, or rename operations on the Layer Edit ***form (Layer Manager) are immediate and cannot be undone using ctrl-Z.***

The View>Layer menu in the network diagram provides access to additional layer operations such as turning on all layers (All On), turning off all layers except the current layer (All On but current), moving an object or group of objects to a different layer (Move to Layer), and color objects by layer. Coloring by layer can also be performed: see "Color by Layer" on page 4-19.

FIGURE 4-1. Layer Edit Window ("Layer Manager")



4.2.2.1 Setting Current Layer

All new network elements created will reside on the ***current layer***. To reset the current layer, use the Layer pull down menu of the layer toolbar to select the layer to be designated as the current layer.

Alternatively, the current layer can be set through the Layer Manager ( ). Click the name of the layer to be set as the current layer and then click Current. The current layer is identified below the buttons on the form and above the table of available layers (the current layer is named "PLATE-top" in Figure 4-1).

4.2.3 Layer Sets

The *Layer Set* tab of the Layer Manager (Figure 4-2) provides a mechanism for creating and activating ***layer sets***, comprised of one or more member layers. A layer set is useful for controlling the visibility (and freeze/thaw) of groups of layers at a time.

When a layer set is activated using the Activate button, *all the member layers of that set are activated and all nonmember layers are inactivated. Any particular layer can be a member of more than one layer set.* Furthermore, multiple layer sets may be selected before pressing the Activate button, activating any members of any selected layer sets.

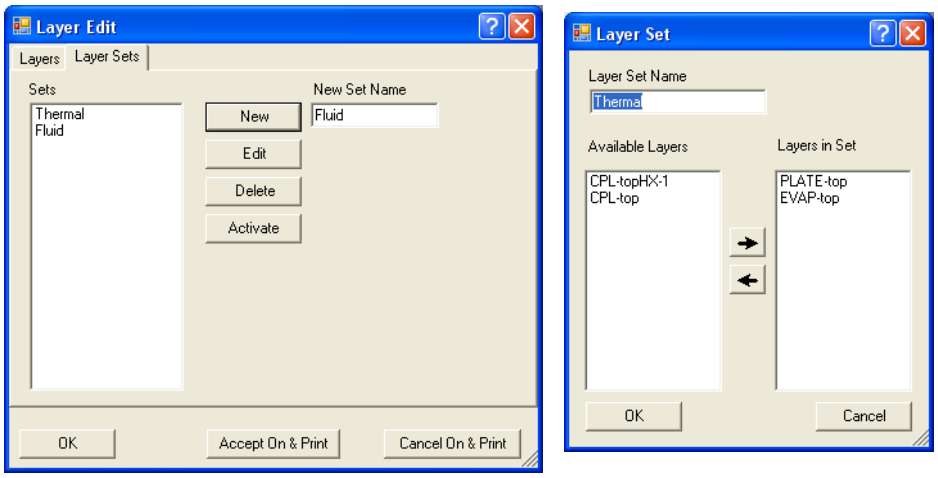
For example, if there are four layers, A, B, C, and D and if they have been grouped into layer sets ABC (consisting of members A, B, and C) and ABD (consisting of layers A, B, and D), then the activating layer set ABC activates layers A and B and C but inactivates layer D. Subsequently activating layer set ABD inactivates layer C and activates layer D, while A and B remain activated. Highlighting both layer sets and activating them both turns all four layers active in this case.

To create a layer set, type in a name in the New Set Name field and click New. A new window will appear with a list of layers on the left. To add a layer to the layer set, click on the name to highlight it and then click the arrow button to move it to the right pane. Likewise, layers can be removed from the layer set by highlighting and using the left arrow button.

Once created, a layer set can be edited to modify members, or it can be deleted. Deleting a layer set does not delete any of the member layers.

After creating layer sets, a set can be activated by highlighting the set name and then clicking the Activate button. The Accept On & Print button will update the layer visibility in the network allowing a visual verification of active layers. Multiple layer sets can be selected and activated at once.

FIGURE 4-2. Layer Set Dialog



4.3 Creating Thermal/fluid Networks

Arbitrary networks can be created using the various network elements available to the user. Since Sinaps is nongeometric, the network is often an abstract representation of the system to be modeled based on assumptions and simplifications deemed appropriate by the user. Thus, any two analysts may arrive at a different network to answer the same design question for the same device or system in question.

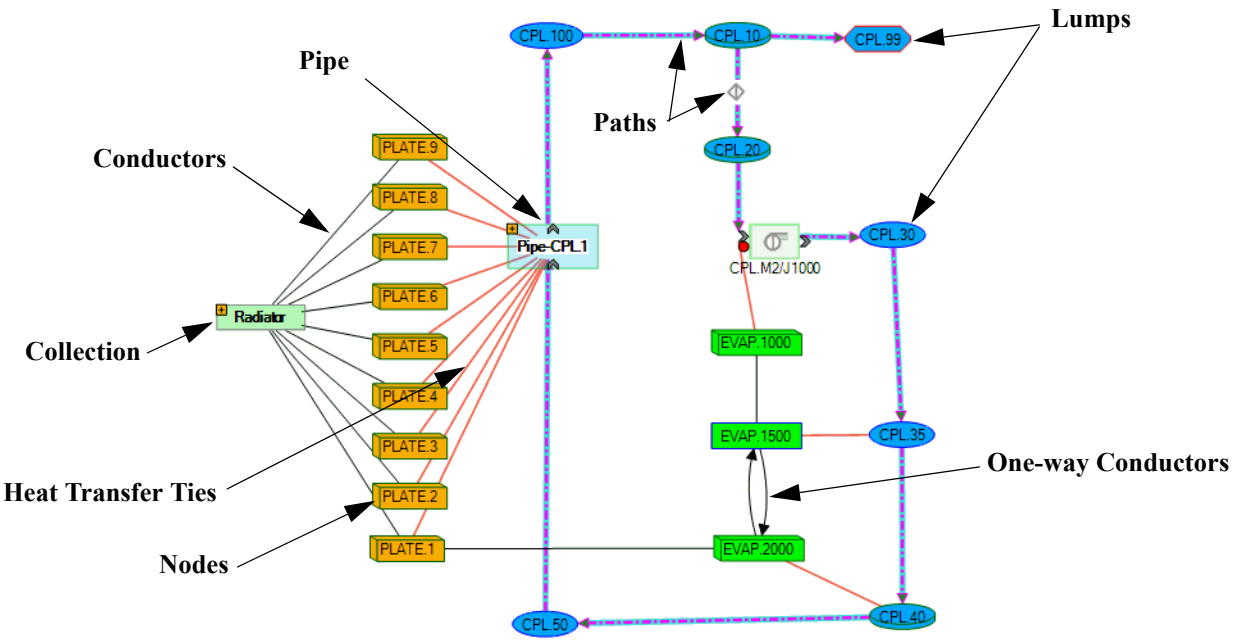
Network elements can be divided into two functional categories, ***objects*** which generally represent thermal mass or fluid volume and are placed on the network diagram by the user, and ***links*** which represent heat transfer or fluid flow connections between any two objects.

The primary objects used in modeling are ***nodes*** representing a mass of solid material, a surface, or a boundary condition within a thermal submodel, or ***lumps*** representing a volume of fluid with a characteristic thermodynamic state, or a flow boundary condition. Links are the connections between objects such as ***conductors***, ***paths***, heat transfer ***ties***, etc. and represent the transfer of heat, mass, or energy. To create a network, the user simply places the appropriate objects on the diagram and links them together with the appropriate connections.[1]

Depending on their type (thermal vs. fluid), objects and links are organized by ***submodel***. A thermal submodel (consisting of nodes and conductors) represents conduction, radiation, and thermal capacitance effects. A fluid submodel (consisting primarily of lumps and paths) represents convection, advection, hydrodynamic effects, etc. The two types of networks may be interconnection via heat transfer ties (which themselves belong to fluid submodels).

Figure 4-3 depicts a typical thermal/fluid network. The network has been colored by layer with each submodel having a unique layer. The submodel PLATE includes a ***collection*** (i.e., a temporarily collapsed subnetwork, Section 4.7) defined as "radiator" which is shown in the collapsed state. When expanded, the collection will show several additional network objects and links. The fluid submodel CPL contains a Pipe which is also shown in a collapsed state: Pipes (which are similar to a FLUINT LINE or HX macros) are always treated as expandable and collapsible collections, whereas other collections may be user-defined. Conductors, paths, and heat transfer ties are each depicted with a unique line type to help visually distinguish the different link types.

FIGURE 4-3. Typical Networks



1 Learning what is "appropriate" requires SINDA/FLUINT training, which is beyond the scope of this text. Refer to that manual, and to the training resources available at www.crtech.com.

4.3.1 Network Objects

The various network objects available to the user are shown in Figure 4-4 depicted as they appear in both the Tool Pallet and toolbars. The object types can be broken into various categories, general drawing objects, thermal, fluid, components (such as pipes), and post-processing.

Using the general drawing group, the user can create comments, or draw lines, boxes and circles in the network for annotation purposes. The user can place these objects anywhere in the network to help with model documentation.

There are three types of thermal nodes: diffusion, arithmetic, and boundary. As shown in Figure 4-4 each node type has a different icon. ***Diffusion nodes*** have finite mass or capacitance: they can store and release energy over time. ***Arithmetic nodes*** have no mass and therefore have no ability to store or release energy: they represent an instantaneous or steady-state solution, even in a transient. ***Boundary nodes*** are a constant temperature condition or infinite capacitance.

Similar to thermal nodes, there are three types of fluid lumps: tanks, junctions, and plena. ***Tanks*** (control volumes) have finite volume and can store energy and mass, therefore resisting changes. ***Junctions*** have no volume and therefore respond instantaneously since they cannot store energy or mass; they represent tanks in the limit of zero volume. ***Plena*** (plural of ***plenum***) are an infinite volume and do not change thermodynamic state. All lumps are characterized by a single thermodynamic state defined by temperature, pressure, and quality.[2]

Fluint Macros are predefined subnetworks recognized by FLUINT (Pipe, CAPPMP). Within Sinaps, the Pipe is used to define either a FLUINT LINE or HX macro (however Sinaps Pipes are more than just SINDA/FLUINT macros, they also create the thermal network associated with the pipe wall.) Macros, like other simpler network objects, can be placed on the diagram and connected into the network. The network associated with a Pipe is expandable, but a CAPPMP has a fixed number of member objects.[3]

Clones. Clones are copies of a node or lump that may be used purely for visual clarity: they do not affect answers. For example, a node with many conductors and ties attached to it can be cloned into two or more copies on the screen, each with its own set of connections. When the model is sent to be solved, only one node will exist in the model itself. (To verify this, try renumbering a clone: all other copies of that node or lump will also be renumbered simultaneously.) When post-processed or colored by results, all clones will have the same color.

To create a clone, click the right-hand mouse button while hovering over the node or lump to be cloned, and select clone from the pop-up list.

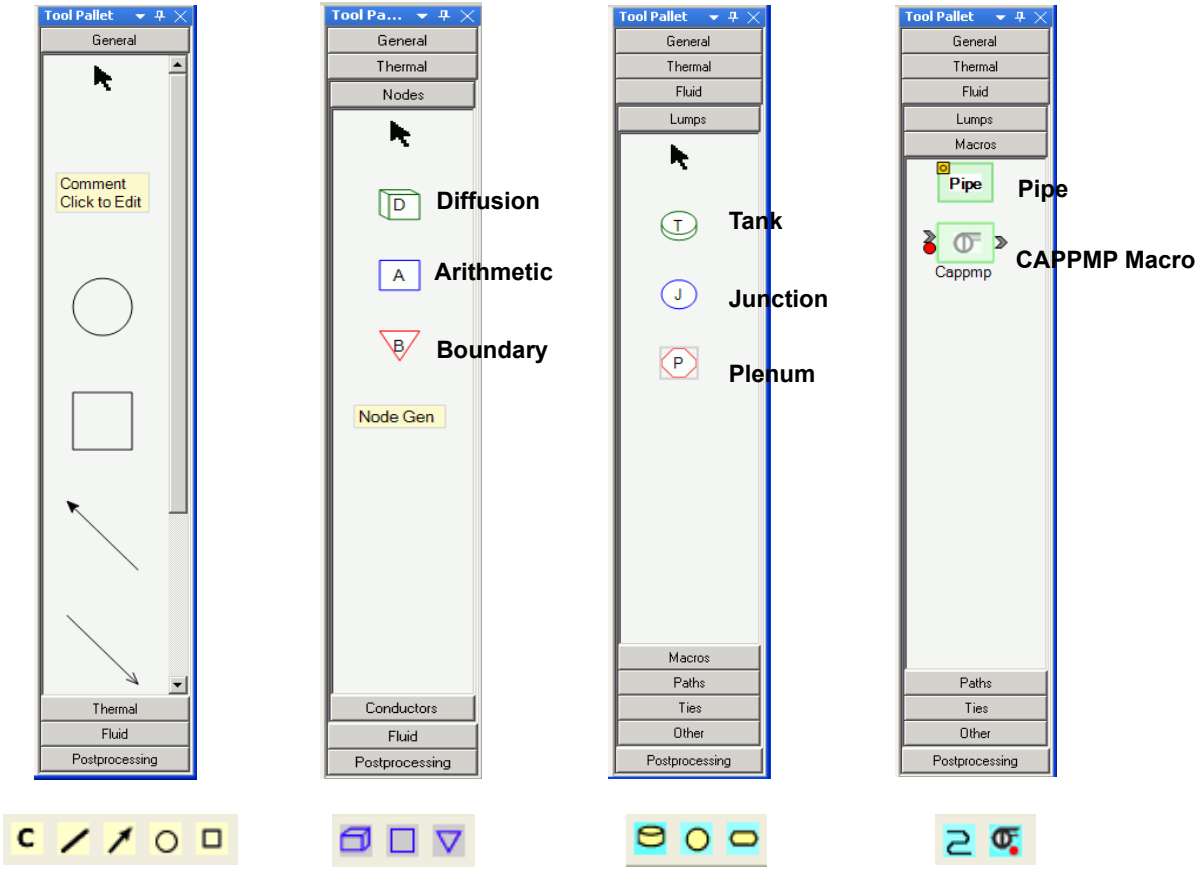
To delete a clone, delete it as if it were any other node or lump: the other instances will not be deleted. In other words, deletion applies only to the selected clone, and not to all instances (copies) of that object.

> Tip: A common use of a clone is to make a single node or lump appear on more than one layer simultaneously. In other words, each clone can be assigned to a distinct layer.

2 Quality is the ratio of gas and/or vapor mass to total mass.

3 CAPPMPs are very specialized elements used to define a capillary evaporating surface: a heated meniscus or section of wick within a device such as a loop heat pipe (LHP), capillary pumped loop (CPL), or in a detailed model of a heat pipe.

FIGURE 4-4. Types of Network Objects



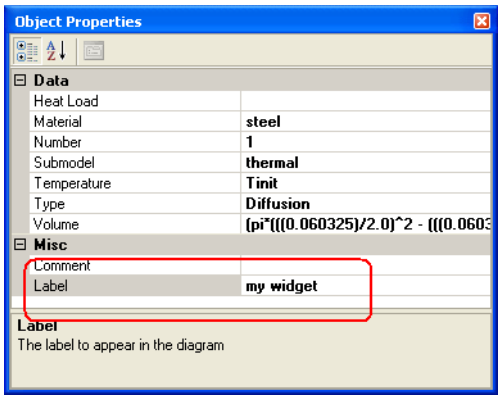
Placing Network Objects. There are two methods for creating objects in a network. The user can click on the icon in the tool bar (or in the tool pallet) and then click on the drawing field to place the object. Alternatively, the user can drag network objects from the tool pallet onto the drawing field. Once placed on the drawing field, the object can be connected to the network with links.

When an object has been selected in the Tool Pallet it will appear highlighted. Locating the cursor in the diagram and then clicking the left mouse button will automatically drop the highlighted object type into the diagram, assigning it to the current layer. Subsequent mouse clicks will add more of the same type objects to the network until either a different object type is selected, or until the *select* mode is enabled. The *select* (vs. *create*) mode is chosen either by clicking the right mouse button, or by clicking on the select cursor icon in the Tool Pallet. The select cursor icon is located at the top of the General, Lump, and Node panels.

Note: Nodes and lumps will be assigned by default to the last appropriate (thermal or fluid) submodel created. The default ("current") submodel can be changed at any time, or the assigned submodel can be edited after an object has been created.

Network Object Label. The label of a network object can be changed using the Object Properties dialog (see page 8). In the case of nodes, conductors, or pipes the user may want to revise a label to be more descriptive. To change the label, use the main Sinaps menu and select View>Properties to open the dialog. Click on the object to be relabeled and the contents of the Object Properties box will update to reflect the selected object. Use the Label field to enter a new label.

Note that the new label will only appear in the network. When postprocessing for text or plotting, the revised label will not be used.



4.3.2 Network Links

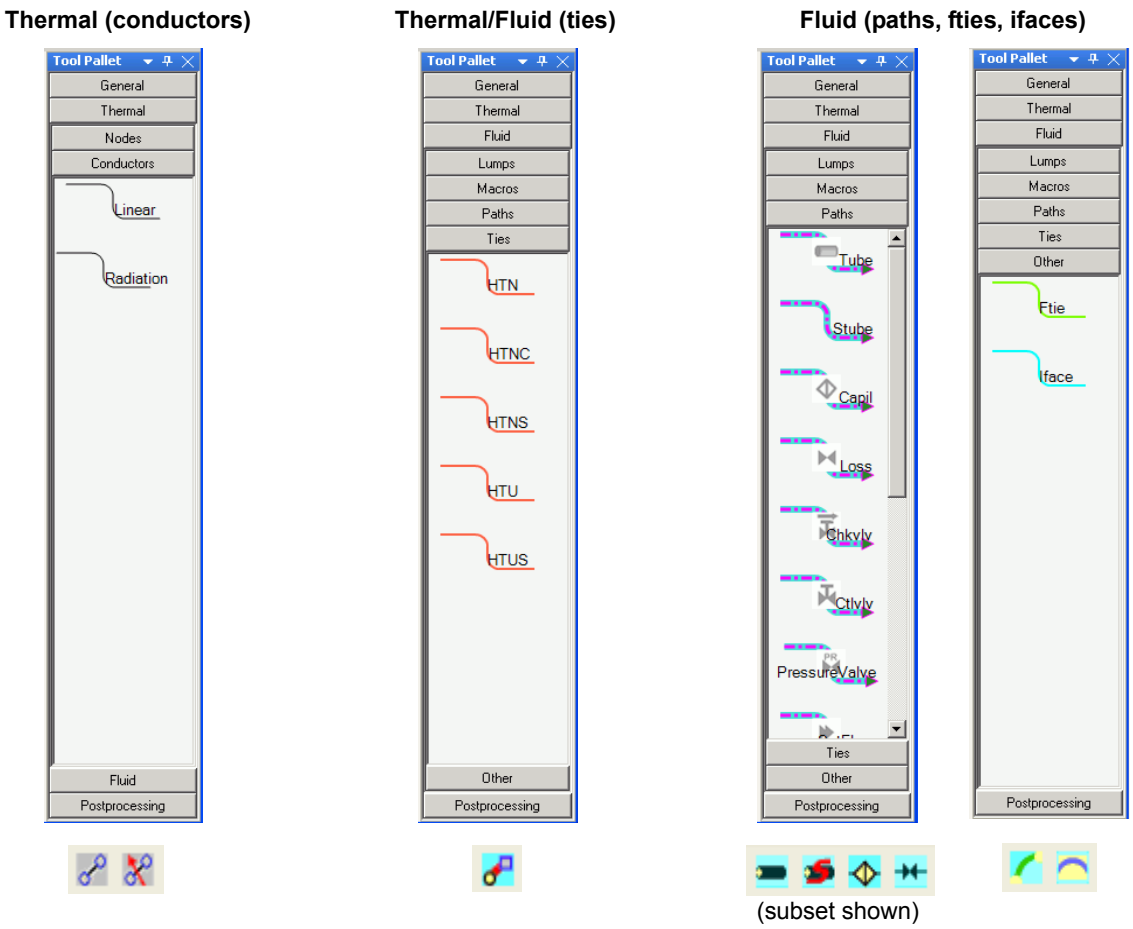
Links between network objects are created by hovering the cursor near the perimeter of an object until the cursor turns into a hand, After this cursor appears, hold the left mouse button down and drag a line to the object to be linked to that first object. Alternatively hover the cursor over the first object, press and hold the shift key while left clicking on the object to activate drawing mode. Hold the shift key as the cursor is pulled away from the object and the link is drawn to the second object.

Sinaps will recognize the object types being connected, and will generate the most recent appropriate link for those objects. Consequently, if a link is created between two nodes, Sinaps will create a conductor. A link between a node and a lump will become a heat transfer tie. A link between two lumps will become an STUBE path unless some other type of path, IFACE, or FTIE has been selected in the Tool Pallet. Like network objects, a link type can be selected and highlighted in the Tool Pallet or toolbar, specifying the type of link to be created next in the network.

> Note: Paths, fluid-ties (FTIEs), and interfaces (IFACEs) are all types of lump-to-lump links of which only one can be selected (or highlighted) in the Tool Pallet.

Links may also be created by dragging them from the Tool Pallet to the Network Diagram, even if they are dragged into a region without any appropriate objects with which to link. The user may then drag the ends of the links to the desired network object. However, this usage is somewhat risky since stranded paths, ties, and conductors will generate errors if they are not connected before a case is launched. Use of the Network Input Check option (“Network Input Checking” on page 4-20) can be used to identify such stranded elements.

FIGURE 4-5. Types of Network Links



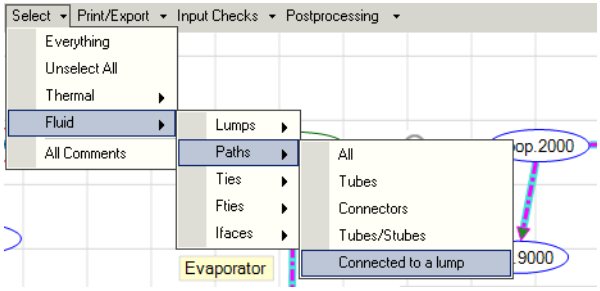
4.4 Selecting Network Elements

When working in the Network Diagram, elements can be selected in one three ways: selecting with the mouse (left click of mouse), windowing a group of elements, or using the Select menu options. These selection methods can be used for moving elements, editing multiple elements, or for network post-processing. To select multiple elements, hold the control key down during the selection process. Note that holding the shift key forces the curser into draw mode for drawing links between objects and may preclude selecting multiple elements. To assist selecting network elements in a Pipe, right clicking on the border line of the pipe provides access to a selection menu to select all items of a specific type within the pipe.

Mouse and window selections methods were discussed in “Mouse Operation” on page 2-2, so this subsection focuses on the Select menu methods.

The Network Diagram Select menu (Figure 4-6) allows the user to select by element types (e.g., lumps, nodes). Element selection can further be filtered by features such subtype (e.g., diffusion versus arithmetic nodes) or connections (e.g., conductors attached to a specific node). For example, due to the difference in magnitude between linear and radiation conductances, linear conductors might be selected for colorization while at the same time radiation conductors are selected separately for thickening based on heat flow.

FIGURE 4-6. Network Selection Menu



To assist with selecting network elements within a Pipe, the user can right click on the frame of the pipe to access the object select menu. This menu provides access to selecting pipe elements by type.

4.5 Editing the Network

Sinaps provides multiple methods for editing network elements: pull-down menus, pop-up menus, edit forms, and the Object Properties window. Details of editing each type of network element differ depending on the type of element, so that discussion is deferred to “Creating Models in Sinaps” on page 5-1. This subsection will instead provide an overview of the different edit methods available in general.

Any single network element in the diagram can be edited by double clicking on the object icon or link to bring up the element edit form. After editing the element through the form, changes are saved and the form is dismissed when the “OK” button is clicked. The edit forms have built-in error checking to verify required data fields are not left blank. If a required field is left blank a message will appear identifying the missing information.

Sinaps supports editing multiple objects at one time. When several items are selected, a multi-edit operation can be invoked by holding the cursor over one of the selected objects and click the right mouse button to access the operate menu. The Thermal and Fluid options in the menu allow the user to select which types of objects are to be edited (nodes, conductors, lumps, etc.). For example clicking Fluid>Edit paths will bring up the path edit form. If more than one type of path has been selected, Sinaps will display the edit form of the first path selected and only the parameter(s) changed will be applied to the selected objects. For example, if tubes and losses are selected and the K-factor is changed, the change will be applied all the paths, however if the length is changed it will only be applied to the tubes since length is not applicable to losses. The network select menu options provide additionally filtering of objects.

Alternatively, all modeling elements can be edited in the ***Object Properties*** task panel that is accessible through the pull down menu View>Properties under the main Sinaps workspace window. The Object Properties is considered an

advanced feature and the user should use it with caution. This panel provides direct access to all commonly used device parameters, which advanced users may find to be more convenient than using the edit forms. Unlike edit forms, the Object Properties panel does not provide input error checking. However, other input checking methods are available as alternatives (see "Input Checking" on page 4-17). The Object Properties panel assumes advanced knowledge of all parameters since the descriptors for the various parameters often correspond to the SINDA/FLUINT text input file acronyms: AF for flow area of a path, DUPI for upstream duplication factor on a path, etc. To help new users associate these acronyms with the parameter names on the edit forms, all edit forms include the acronyms in parentheses after each parameter description.

The Network Diagram window supports traditional windows-style commands such as copy, paste, cut, and undo (ctrl-c, ctrl-v, ctrl-x, ctrl-z respectively). Copy and paste commands work for all network elements (nodes, lumps, and pipes, comments, etc.) however links (paths, conductors, etc.) can only be copied as part of a selected group. A group of network objects and links can be copied, but only the objects and *contained* links (meaning only links that are between members of the selection set) can be pasted back onto the network. Links between the group of objects being copied and the rest of the network will not be copied.

Tip: If you need to copy a single link (conductor, path, etc.), the set as default option can be used. Right click on the link and select "Set as default" and then draw new links in the network. It is recommended the default be "Reset" after using the set as default option since defaults are saved with the model and will be carried over from session to session.

Tip: The group copy option can be used to copy a selected group of objects and contained links within a submodel, across submodels, or across two master model files. Used in conjuction with dereferenced network logic, this copy feature will allow a "simple" component to be copied from one model to a second model. Such a copy paste action is limited to one thermal submodel and one fluid submodel. Objects will be pasted into the "current" thermal and fluid submodel. Note the paste operation is limited to one thermal submodel and one fluid submodel during. Pasting of more than one thermal or one fluid submodel will result in everthing being pasted into the current submodle.

Note: When copying an entire submodel, it may be easier to use the submodel export/import feature, see "Submodel Import and Export" on page 11-5. This will capture logic blocks, control data, array data, etc. in addition to the network.

An alternative method for copying network objects is to hold the control key while left clicking on an object, dragging off an object and releasing the control key. This will copy an object or group of objects but *not* the links between objects. It also will not copy pipes, collections, or postprocessing scales. However it will let you copy a single or group of objects in a pipe however the new objects will not be part of the pipe.

Certain activities cannot "undone" (using ctrl-Z) such as object editing through the object properties task panel, editing through an object edit form, pasting of pipes, etc. Sinaps will clear the undo buffer after any of these restrictive actions are performed. Hitting "undo" after such an event will result in a notice stating the buffer is empty, indicating the action cannot be undone.

4.6 Network View Options

The View menu in the Network Diagram provides access to zoom options, an overview feature, and layer editing/ options. The zoom and overview visualization options will be described in this subsection. (The Layer Manager is described in Section 4.2.)

This section will also describe panning options, as well as global network layout options.

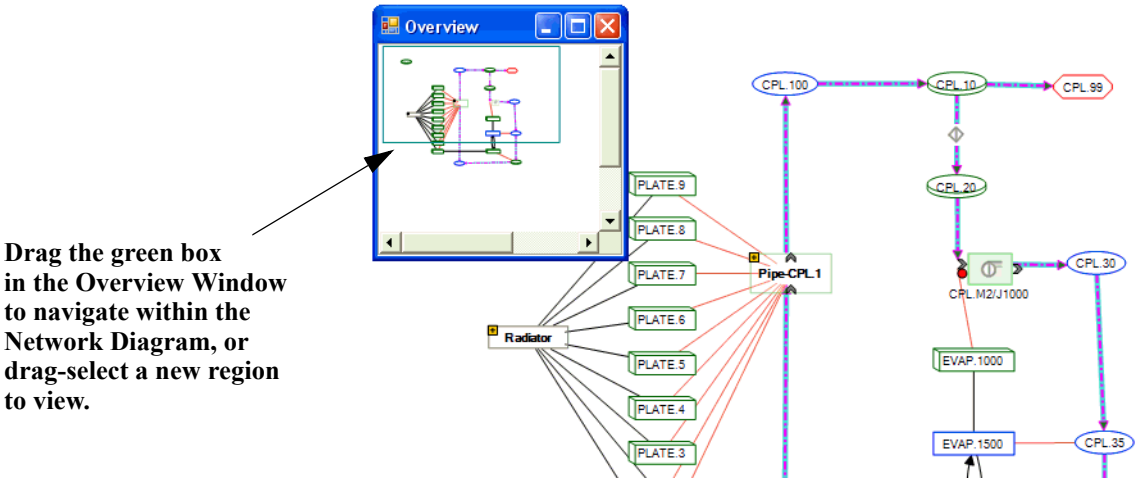
4.6.1 Network Zoom Options

The View>Zoom In and View>Zoom Out options allow the user to resize everything within the Network Diagram, and View>Zoom Normal returns to the standard size. View>Zoom to Fit chooses a zoom factor such that all visible elements are contained within the current Network Diagram window size.

4.6.2 Network Overview

When View>Overview is selected, a new navigation window (shown in Figure 4-7) will open providing an overview of the entire network. The user can quickly modify which portion of the network is shown in the Network Diagram window by dragging the green rectangle in the Overview window. This is very useful when working with large networks and works in lieu of pan and zoom features.

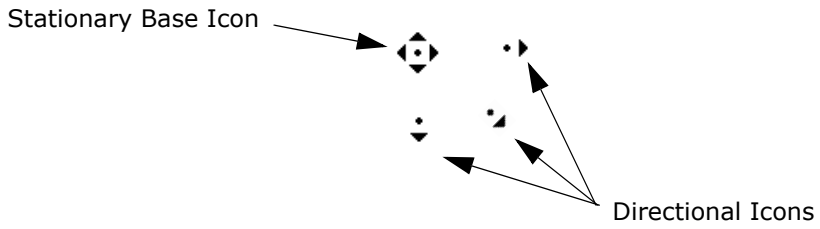
FIGURE 4-7. Network Overview



4.6.3 Panning through the Network

Sinaps allows the user to pan through the network if a three-button mouse is available. Simply click the middle mouse button and the pan base icon will appear on the network. As the cursor is moved away from a stationary icon, a directional icon will appear. As the cursor is moved around the diagram the panning direction will changed as the directional icon pivots around the stationary icon. Panning can be performed in 45 degree increments around the base icon. To exit pan mode, simply click the right mouse button.

FIGURE 4-8. Network Pan Operations



4.6.4 Link Labels

Link labels can be toggled on or off. To toggle an individual link label on or off, access the object edit menu by right clicking on the link and selecting Toggle Label. All the link labels in a network can be toggled on/off using the network window menu option View>Link Label Toggle.

4.6.5 Network Layout Options

The Layout menu provides access to toggling a grid on/off, working with collections (described in Section 4.7), aligning objects, and creating automatic layouts.

By default, object locations within the diagram are arbitrary. By selecting Layout>Align to Grid, when objects are next moved, they will only be allowed to reside at grid points (visible using Layout>Show Grid), resulting in an orderly network. Toggling Layout>Align to Grid returns the diagram to the normal free-form object location mode. The grid size can be changed as described in “Network Field Properties” on page 4-21.

Various object alignment options are available. To use them, first select the objects to be aligned or otherwise redistributed. As an example of these otherwise self-explanatory options, Layout>Align Left Sides will move all selected objects horizontally to the left-most location of any selected object. Layout>Even Space Horizontal and Layout>Even Space Vertical are particularly helpful for redistributing objects into a visually clear and orderly pattern.

The Automatic Layout options include layered digraph and force directed.[4] The *automatic layout option is primarily used when reading a model in from a text file*, since such models have no graphical basis for the depiction of the net-

4 A “force directed” graph is viewed as a system of bodies with forces acting between the bodies. The algorithm seeks a configuration of the bodies (i.e., network objects) with locally minimal energy, i.e., a position such that the sum of the forces on each body is zero.
The “layered digraph” uses a hierarchical approach for creating drawings of directed graphs with network objects arranged in layers. The term “layer” in this context does not refer to Sinaps Network Diagram layers, but rather to sheets of heavily interconnected objects.

work. Nonetheless, if the network has grown too cluttered, it can be automatically redrawn using these options as well, remembering that ctrl-Z will return the objects to their original locations if the automatic methods are not helpful.

> Note: Automatic Layout currently applies globally to the entire network, and not just to the current selection set.

Sinaps will automatically use the layered digraph option for creating an initial network for a model that has been read in.[5] Afterwards, the user can toggle the layout to forced direct to see if that alternative offers a more appropriate starting point for further customizing.

4.7 Collections

To aid in network organization, the user can create a ***collection***: a group of objects specified as a subnetwork that can be collapsed into a single icon, and re-expanded as needed. A collection is essentially a place-holder for network objects (including clones) and their associated connectors.

A collection can contain any number of network objects or comments. The only restriction in creating a collection is that all objects selected to be grouped into the collection *must* reside on the same layer. If objects do not reside on the same layer, Sinaps will move them into the current layer when making a collection.[6] Multiple collections may be nested: a collection may be added as a member of a new (higher level) connection.

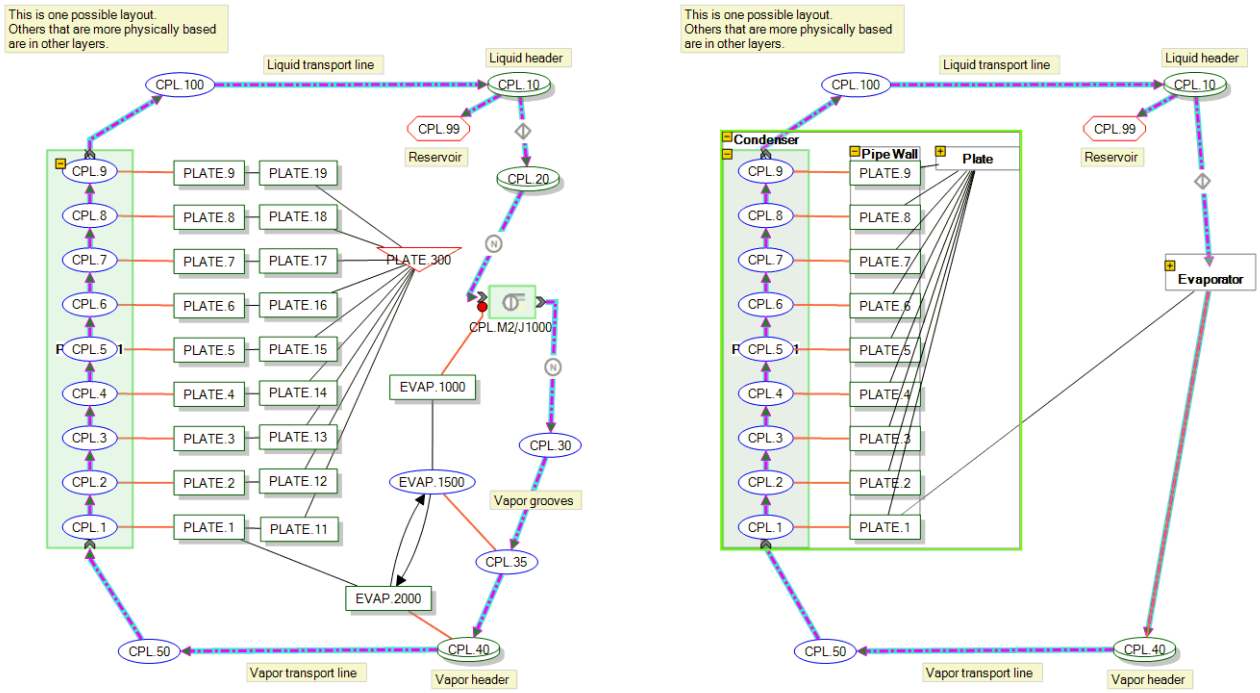
Figure 4-9 shows an example of using collections to simplify a network. The original network shown on the left has been simplified with the use of collections as shown on the right. The nodes in the condenser portion have been grouped into two collections representing the *Pipe Wall* and the *Plate*. In the view at the right, the *Pipe Wall* collection has been expanded while the *Plate* collection has been collapsed. These two collections were in turn nested with the Pipe "CPL" into a collection labeled *Condenser.* The final collection in the example is labeled *Evaporator* (at the right of the right-hand diagram). The *Evaporator* collection contains portions of both the thermal and the fluid which represent the evaporator.

To select a collection, click on the yellow select button in the upper left corner of the collection. After selecting a collection, it can be moved to a new location on the network, moved to a new layer, or deleted.

5 For huge models, the cost of a generating layered digraph can be excessive. In order to avoid the need to abort model read-in, Sinaps attempts to auto-detect this condition and, if encountered, it will resort to a simple "raster scan" approach to object placement. The user may then attempt a force directed graph as an alternative, since the cost of that layout option is usually smaller.

6 Consider cloning objects if there is a need to preserve the layer designation for the original object; clones of a single object may reside on distinct layers and within different collections.

FIGURE 4-9. Example Usage of Collections



To create a collection: Select the network objects to be placed into the collection and then use the pull down menu in the network window Layout>Collections>Create from Selections. A green box will appear around the group of objects designating it as a collection. The default label “Collection” will appear in the upper left corner of the box.

Selecting Network Elements in a Collection: Elements within a collection can be selected individually or in groups for editing, moving, etc. To assist in group selection, Sinaps provides a means to select all elements of a certain type in a collection. Place the cursor over the yellow select button in the upper left corner of the collection and click the right mouse button to access the object edit menu. The action list includes a select option which will select all lumps, nodes, paths, etc.

Renaming a collection: The default name for a collection is “Collection.” The collection can be renamed by clicking on the label, which activates a text edit box.

Moving objects into or out of a collection: Additional objects may be moved into a collection by highlighting the object(s) and dragging them into the bounding box of an open (expanded) collection. Objects can be removed from a collection by pressing and holding the ctrl and alt keys together while left clicking and dragging and object out of a collection.

Emptying/Deleting a collection: To delete the collection, it must be emptied after which it will automatically disappear. To empty a collection, highlight the collection and choose Layout>Collections>Empty from the Network Diagram pull-down menu.

> Note: Highlighting the collection and then hitting the delete key or selecting Edit>Delete from the menus will delete the collection and all objects within it. Use ctrl-Z to undo this action if it happens accidentally.

4.8 Network Printing and Export

The Print/Export menu option allows the user to print and export the network diagram.

The Print All and Print View are toggle options for printing either the entire network or only the portion of the network that is currently displayed on the screen. After selecting either of these toggles, the user can either print the selection or export it. The print size is fixed, and may extend beyond one page.

The export feature allows the user to export the network as a graphic image in either jpg, gif, png, bmp, tiff, or wmf formats.

> Note: Background images are not exported along with the network.

4.9 Input Checking

Sinaps provides a variety of input checks: 1) visual preprocessing of the network, 2) checking for missing inputs, and 3) visual verification of automatic macro generation.

4.9.1 Visual Preprocessing

The visual preprocessing feature is similar to post-processing. It allows the user to verify initial conditions specified in the model. The Input Check menu allows the user to color nodes, lumps, conductors, paths, ties, fties or ifaces in the network. By default each type of element will be preprocessed based on the parameter designated in the current hover text settings (see “Hover Text” on page 3-12). Parameters can be set using the menu option Input Checks>Set Parameters.

Alternatively, links can be thickened and objects/links can be colored by material. Using the Select menu, prior to performing an input check, the user can select all or any portion of the model for checking. If nothing is selected, or if only one item is selected, then the entire network will be checked. Unlike post-processing options, which color data from

save files, the input check option colors based on initial values only, and therefore does not require the save file or starting time to be specified.

> Note: Since some object parameters may be defined based on processor variables which are unknown prior to a solution, be aware that such initial values can skew the color scale.

After checking inputs, color bars and thickening scales will be displayed on the network. These scales can be moved simply by placing the mouse over the scale, click the left mouse button to highlight it, and then drag the object to the desired location. Thickening scales can be moved in the same fashion.

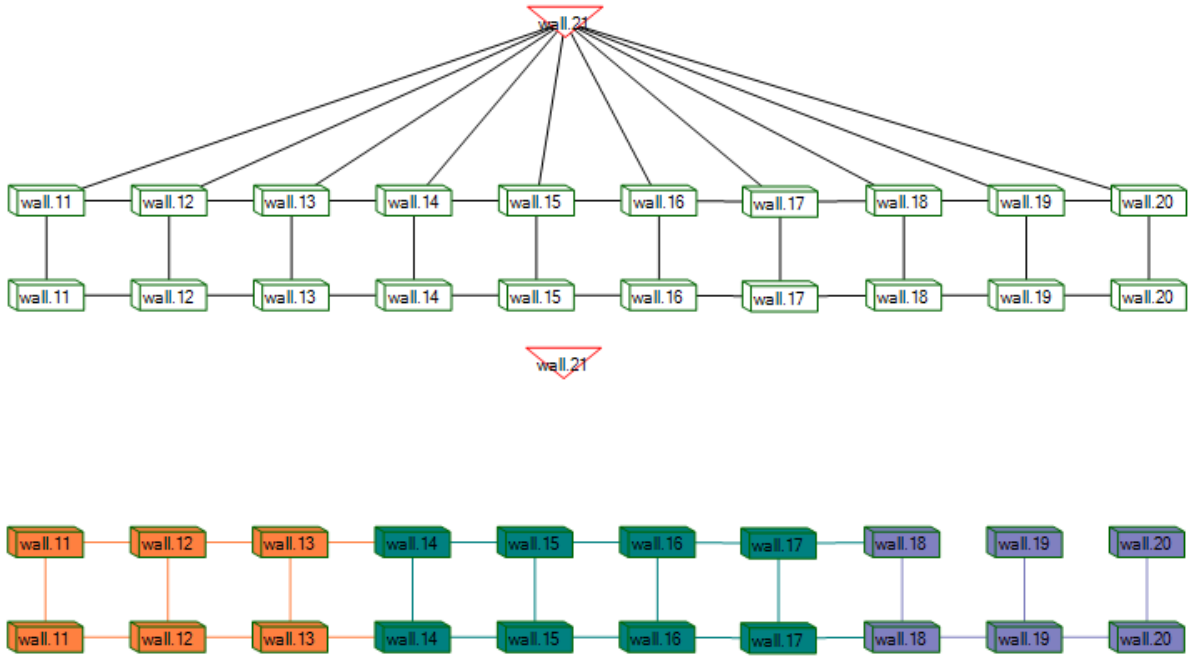
Editing the Color Bar or Thickening Scale: The color bar is editable through the Object Properties panel, which is accessible using View>Properties on the Sinaps main workspace window pull-down menu. Simply highlight the bar or scale and the associated parameters will display in the Object Properties panel. The user can adjust the resolution, autoscale/min/max, title, and orientation (color bars only). For more information on customizing scales, see Section 10.4.2.1.

4.9.2 Color by Material

To provide a visual verification of how thermophysical properties are applied in a model, the network can be colored by material. This option will color all nodes and conductors which are based on a material defined in the thermophysical properties database. Each material will display as a unique color. The color gray indicates the node references a material not present in the material database.

To color by material, select the group of objects to check (selecting nothing will default to checking everything) and choose the Network Diagram pull-down menu option Input Checks>Color by Material. Figure 4-10 depicts a network before (top) and after (bottom) being colored by material

FIGURE 4-10. Color by Material Example



4.9.3 Color by Layer

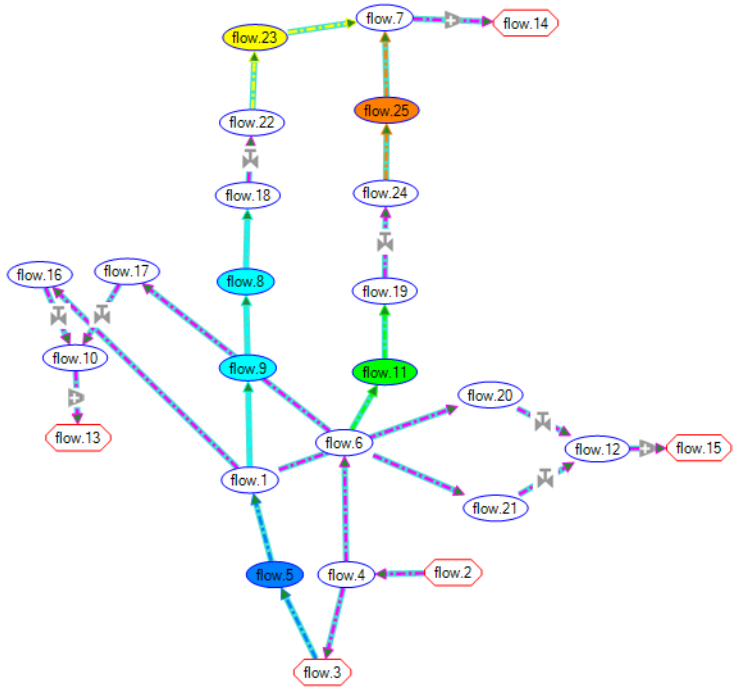
The network can be colored by layers to identify which object reside on which layer. Using the Network Diagram pull-down menu option View>Layer>Color>Color by Layer will result in all network objects being colored. Alternatively, click on the Color By Layer icon (.) on the Layer toolbar. Objects with similar color are located on the same layer. Links between objects are not colored since they are not assigned to any specific layer.

4.9.4 Checking for Created Ducts

Any series of individually created lumps and paths that Sinaps has identified to be a continuous flow passage is flagged internally as duct, or a SINDA/FLUINT Line macro. The user can inspect the network to identify all "created ducts" using the pull-down menu option Input Check>Show Created Ducts. Each duct found will be identified by a different color on the network. Both the lumps and the paths included in the duct will be colored. In the example depicted in Figure 4-11, five discrete ducts were identified and created.

For more information on automatic ducts see "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5.

FIGURE 4-11. Visual Check of Automatically Created Ducts



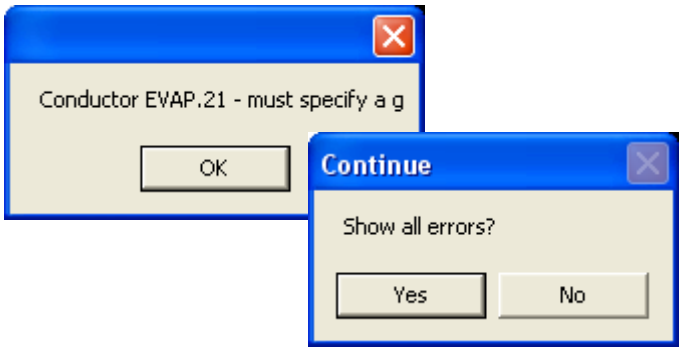
4.9.5 Network Input Checking

The network input checks can be invoked using the pull down menu Input Checks>Network Check. The feature will step through the entire network looking for missing inputs (required parameters which have been inadvertently left blank) and orphaned network elements. All missing inputs will be flagged to the user via a pop-up window as shown in Figure 4-12. If more than one error is detected, the user will be given the option of viewing all errors. If the user opts to view all errors, a text window titled “error list” will open listing all errors found during the input check.

If no errors are detected, Sinaps will state no errors were found during the simple check and ask if the user wishes to run the preprocessor for further error checking. Running the preprocessor will invoke the spell checker along with identifying syntax errors in logic, and verify the include file paths. If an error is found during the preprocessor, a window will pop up at the detection of the first error and terminate the preprocessor.

Alternatively the user can run the full preprocessor from the case manager (see “To Run a Case:” on page 9-3. This option will run the preprocessor through to completion and open the preprocessor output file (pp.out). The user can then search the preprocessor output file for errors.

FIGURE 4-12. **Network Input Checking**



4.9.6 Highlight Associated Paths for Ties

An option is provided to highlight the paths associated with any fluid tie (heat transfer between a lump and node). To access thisfeature, right click on any fluid tie and click Highlight Paths. A window will appear allowing the selection of the highlight color and line thickness. After clicking OK, the paths will highlight. Colors and thickness can be reset using the Network menu option Input Checks>Reset.

4.10 Customizing of Network Field and Elements

Sinaps allows the user to customize much of the interface. Just like the Object Properties task panel can be used to define modeling elements, it can also be used to customize the appearance of the network and all modeling elements.

4.10.1 Network Field Properties

The network properties (grid size, color, etc.) can be customized by the user. To customize the network, simply right-click anywhere on the network field and open the Object Properties panel. The following are examples of customizing that may be performed:

- Change background color
- Add a background image. This is currently limited to a single image positioned in the upper left of the diagram. Once placed on the diagram the graphic will be locked in place and the network can be shifted as appropriate on the graphic.
- Clear a background image
- Modifying the grid size, color, and appearance
- Primary/secondary selection color. The selection color is the color by which an object will be highlighted when selected. *Primary* refers to the first object selected, *secondary* refers to all subsequent objects selected while holding the shift key.

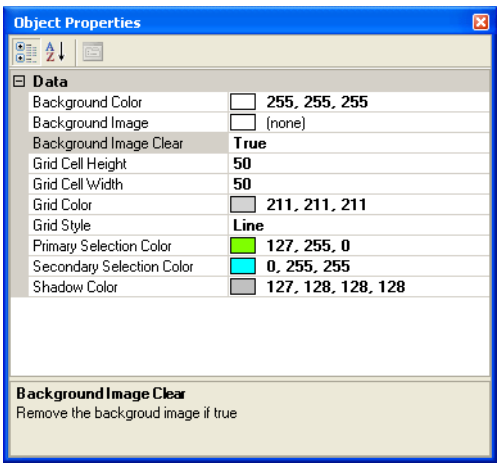
- Drop shadow color. By default, twinned elements are displayed with drop shadows. The drop shadow option controls the color of the shadow.

4.10.1.1 Background Images in Networks

Background images can be added to a network as a visual aide. To add a background image, click the right mouse button over an empty area of the network. The Object Properties window will appear if not already open and the window will display a list of customizable properties for the network as shown below. Click in the Background Image field and a browser icon will appear. Use this browse icon to locate the image to place in the background. The image must be a bmp, gif, jpg, jpeg, png, ico, emf or wmf format. Only one image can be place in the background of the network. To remove the image, set the Background Image Clear field to True.

Once an image has been defined in a network, the image cannot be moved relative to the network. However the network elements can be moved relative to the image.

To create multiple images in the background of the network, a collage of images can be created in a graphics package (MS Paint, Photoshop, etc.) external to Sinaps and the resulting image can be imported as a background for the network. To assist in sizing an image, it is recommended to use the network grid to “measure” the network. Using the network properties dialog shown below, set the Grid Style to Line. The grid cell height and width are defined in pixels. Count the grid lines to determine an optimum size for the background graphic.



4.10.2 Customizing Appearance of Network Elements

(Future releases of Sinaps will allow more extensive customizations, including the appearance of network objects and user-created objects.)

CHAPTER 5

Creating Models in Sinaps

The Sinaps workspace provides a graphical, non-geometric, interface for creating thermal/fluid models in formats compatible with SINDA/FLUINT. A model is typically comprised of network elements, user-defined variables called *registers*, user-defined concurrently executed logic, fluid and thermophysical property data, control data, and optional user-defined subroutines. This section will cover how to create models in Sinaps using the various network elements, register spreadsheet, and user logic.

5.1 Master Model Components

When a model is created in Sinaps, the user is creating a ***master model*** typically comprised of several *submodels*. A ***submodel*** can be thought of as subset of the master model with its own control data and logic. Often submodels represent a component or group of components in a large system, but they can also used for organizational reasons. Each submodel has a unique numbering scheme so the user does not need to worry about numbering conflicts between two or more submodels.

How submodels are defined depends on whether they represent a thermal network or fluid network. The ***configuration*** is defined as a group of submodels contained within the master model which have been selected by the user for inclusion in any single analysis case. Configurations are defined using *build* statements (see “Case Data - Build” on page 9-10).

> Note: No single submodel may have attributes of both a thermal and a fluid submodel; a coupled thermal-fluid analysis must therefore have at least one submodel of each type. Otherwise, a model can consist of one or more thermal submodels, or one or more fluid submodels.

To understand the hierarchy of the master model, the user must have an solid understanding of global- versus submodel-level inputs. It is recommended that the user review Section 2.2 of the SINDA/FLUINT User’s Manual. Certain data blocks such as Operations, Registers, Global Control, Subroutines, and the Advanced Design modules are applicable at the *master* (global) model level; they are based on a single definition that applies to *all* submodels. Other data

blocks such as Node, Conductor, Flow Data, etc. can vary with *each* submodel, and the available *types* of blocks depends on whether the submodel is a thermal or a fluid submodel. Figure 5-1 graphically depicts this hierarchy.

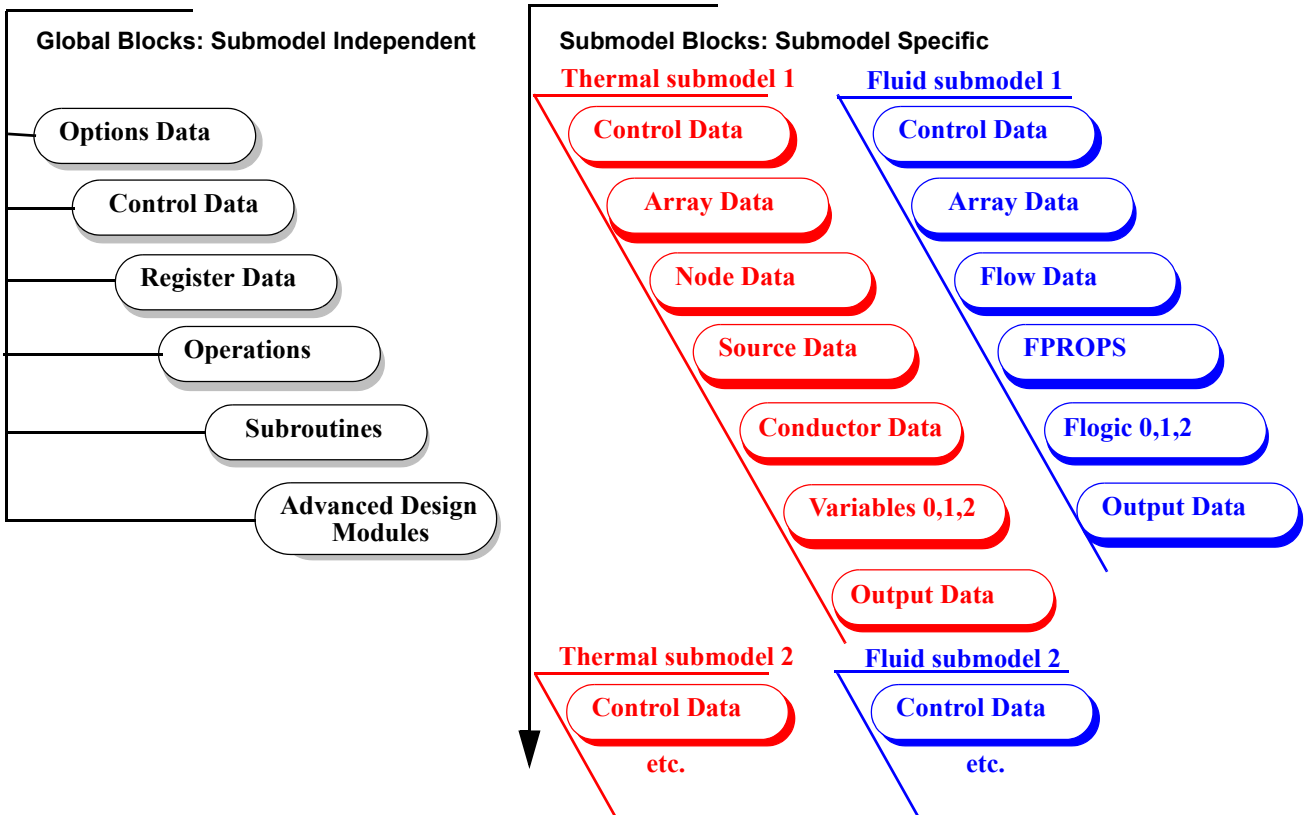
A master model can contain only one Operations block (since it is global in scope), but may contain several Node and Flow Data blocks.

Some types of blocks exist at *both* the global and submodel levels. One such block is the Control Data block. The master model has only one set of control constants* ("Control Data, Global"), and these controls will apply to all submodels by default. However, each submodel could optionally include its own Control Data block which supersedes the global Control Data, and which is applicable to that submodel only. In other words, submodel-level control constants default to the global values unless the user specifies them to be different.

Sinaps provides access to the data blocks in various ways. Node, Conductor, Source, and Flow Data blocks are all created through the network. Register data is created from the register table ("Global Registers"), though values can be overridden per run in the Advanced Tab of the Case Manager (see "Using the Case Manager" on page 9-1). Tables provide the ability to define data in a variety of formats which translate to SINDA/FLUINT arrays. Options, Control, Operations, and Output blocks are all generated by Sinaps according to user settings in the Case Manager while Subroutines, advanced design modules, Variables logic blocks and Flow logic blocks are all accessible through the Advanced Tab of the Case Manager ("Case Data - Advanced Options" on page 9-13).

* These are often referred to as "control constants" even though the term *constants* is a historical misnomer. These parameters can be varied between cases, and often during a run. Therefore, "control variables" would have been a better designation.

FIGURE 5-1. **Master Model Hierarchy of Input Blocks**



When a model is launched (clicking “Run” in the Case Manager), Sinaps creates a text-based input file (*.sin) to be launched in SINDA/FLUINT. The text file will be formatted using the appropriate structure for the solution engine. For SINDA/FLUINT all node data will be written to HEADER NODE DATA, [SUBMODEL NAME] blocks, conductor data to conductor data blocks etc. as defined by the hierarchy shown in Figure 5-1.

To view the generated input for a particular node, search within the text file for the appropriate node header block for the submodel in which the node resides. In the example below, node #20 in submodel LINES has been initialized to a temperature defined by the register *Tinit*. It has a capacitance defined by register *plCap*. It also has a heat load define as the register *power* as seen by the HEADER SOURCE DATA block.

```
HEADER NODE DATA, lines
       2,Tinit,-1.0
       407,Tinit,-1.0
       20,Tinit,plCap
       1,Tinit,ccCap
       10,Tinit,evpCap
       200,Tinit,-1.0
       -999,Tamb,0.0
HEADER SOURCE DATA, lines
       20 = power
HEADER CONDUCTOR DATA, lines
       -151,101,999,TotLen*pi*(ThkVap+Dvap)*EambVp/ResVap
       1,10,1,GEvpCC
```

```
        2,1,999,GAmbCC
        460,20,10,5000*evplen*evpwide
(... etc.)
```

5.2 Creating a New Model

The first step in creating a model in Sinaps is opening the Sinaps application through the Start menu in Windows, such as by choosing Start>All Programs>Sinaps>Sinaps, or by clicking a desktop shortcut if available. Once the application is opened, a new model needs to be created either by clicking on the NEW icon on tool bar or using the pull-down menus and selecting File>New>SINDA/FLUINT Model. A window will appear asking the user to define the ***working directory***, i.e. the directory in which the model file and all supporting files will reside.

The model file (*.smdl) contains almost all the information in the model, such as network information, user logic, registers, and Case Manager information. Two exceptions include material properties (if the Thermophysical Properties dialog was used) and the fluid property (FPROP DATA) information (if an INSERT or INCLUDE file were used instead of a built-in description). These auxiliary files are intentionally excluded since they are often shared between multiple models and usually stored in a central location.

> Note: In order to transmit a Sinaps model to someone, you need to provide the model file (*.smdl) along with the thermophysical properties database (*.tdp) as well as any fluid property include files (*.inc).

5.2.1 Initializing a Model

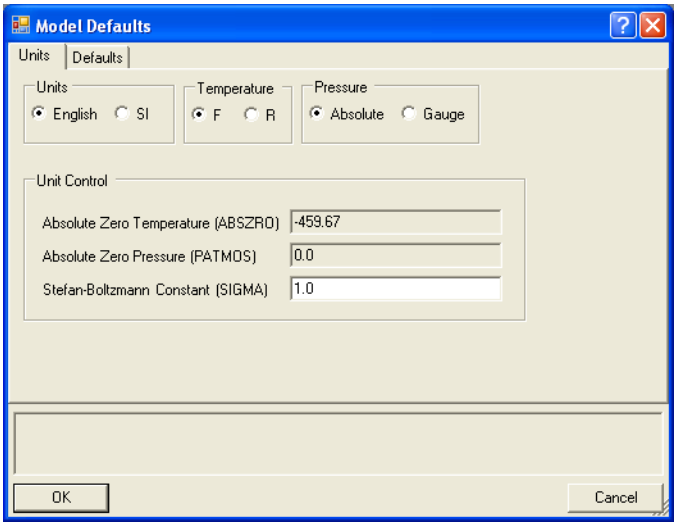
Upon creating a new model, the user will be prompted to define working units, to set network element defaults, and to create submodels. After the initial default dialog, the user can modify units through the Case Manager. The network element default dialog can be accessed through the main pull down menu Model>Defaults. Submodels can be added or deleted using the Model Browser, the Model toolbar, or the menu Model>Add and Model>Delete.

5.2.1.1 Setting Units for the Model

The Units tab shown below allows the user to specify working units as either ENGLISH or SI. The user must also select a preferred temperature and pressure scale. This tab also contains default values for the absolute zero reference temperature and pressure. Table 1 summarized the units for both the ENGLISH and SI options.

When building a model, the user must ensure all inputs and user logic are defined in the appropriate units. To assist the user in maintaining consistent inputs, built-in conversion constants are provided for commonly used conversion. These constants are a subset of the built-in registers (see “Parameterizing a Model” on page 6-1.

FIGURE 5-2. **Setting Model Units**



Setting the Stefan-Boltzmann Constant: The Stefan-Boltzmann constant (σ or ***SIGMA***) is used during radiation conductor calculations. It is a multiplier used to calculate the heat flow through a radiation conductor:

$$Q_{1\text{-}2} = s\ e\ A\ (T_1^4\text{-}T_2^4) \quad \text{or} \quad Q_{1\text{-}2} = s\ F\ A\ (T_1^4\text{-}T_2^4)$$

By default, SIGMA is set to a value of 1.0 indicating that radiation conductors defined by the user already contain the value of the constant. If this is not the case, *the user must define the value of the constant, which will then be globally applied to all radiation conductances.* A convenient method for doing this is to use the built-in constants SBCON and SBCONSI (based in ENG or SI units respectively). These constant names can be typed directly in the data entry field.

SBCON = 0.1712e-8 BTU/hr-ft^2-R^4
SBCONSI = 5.6693e-8 W/m2-K^4

Note: A common mistake is to either include the Stefan-Boltzmann constant twice (both in SIGMA and in the G value of the radiation conductor), or to forget to include it (i.e., let SIGMA default to 1.0 and neglect to include σ in the conductance value). Either way, the radiation conductance will be incorrect by many orders of magnitude, and the software is often unable to trap such mistakes given that radiation conductances do typically vary by a significant amount. Therefore, extra diligence is required by the user to prevent this mistake.

Modifying Units. The current unit set is displayed on the bottom of the General tab (see the lower portion of Figure 9-2, for example). The units are those specified by the user during the initial model creation dialog. If the model was imported, the units will be based on the control constants of the imported model.

When a model is first created, the user is prompted to set units through the model initialization dialog. The results of this dialog are used to set the units on the General tab and in Global Control Data on the Advanced tab. The user must be consistent with the selected unit system throughout the entire model including network inputs, registers, logic blocks and include files (FPROP data blocks, MIXTURE blocks, and thermophysical properties are exceptions).

The user can change the model units using the pull down menu option Model>Defaults & Units. *However, please be aware that inputs including logic, network element inputs, tables, control constants, and registers will* ***not*** *be automatically converted when a change is made to model units.The burden of propagating the changes falls on the user.*

Default Units: Default working units in Sinaps are summarized below:

TABLE 1. Unit Systems, Assumed Units

Parameter	SI (Metric) Units	English Units
Length	meter	feet
Mass	kg	lb_m
Time	second	hour
Temperature	C, K	F, R
Pressure	Pa (pascal)	psia
Specific Enthalpy	J/kg	BTU/lb_m
Energy	J/kg	BTU/lb_m
Specific Heat	J/kg-K	BTU/lb_m-R
Area	m^2	ft^2
Volume	m^3	ft^3
Density	kg/m^3	lb_m/ft^3
Flowrate	kg/sec	lb_m/hr
Viscosity (Dynamic)	kg/m-sec	lb_m/ft-hr
Conductivity	W/m-K	BTU/hr-ft-R
Surface Tension	N/m	lb_f/ft
Gas Constant	J/kg-K	ft-lb_f/lb_m-R

5.2.1.2 Setting Network Element Defaults

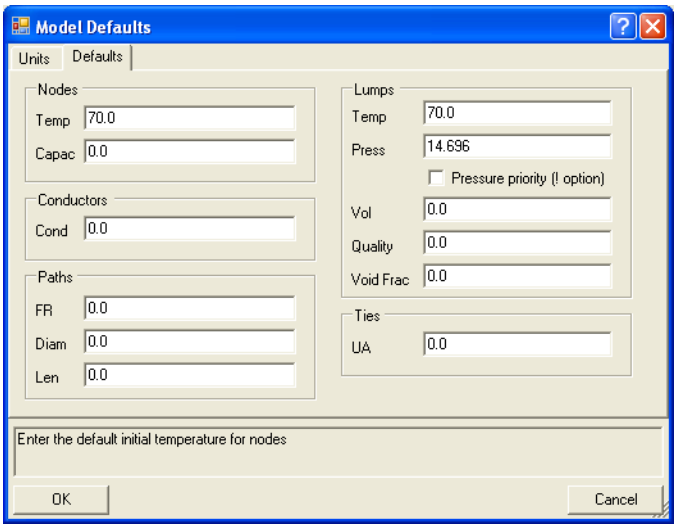
When a new model is started, the user will automatically be prompted to define network element defaults. Once a model has been created, the user can return to the Model Defaults form by selecting the menu option Model>Defaults. Alternatively the defaults can be set via the Model Browser (see “Setting Network Element Defaults” on page 8-5). Note that the Model Browser provides access to all parameters associated with a network element type whereas the Model Defaults form only provides access to to most commonly use parameters. All network element defaults must be defined in the designated working units and may be defined using registers and expressions.

All default fields need not be filled in: some may be left blank. If no value is given for the conductance of a conductor, for example, the missing input will be identified later when if the user performs an input check or launches the run. On the other hand, setting the default to 0.0 may result in conductors with a potentially incorrect value not being flagged when the model is launched. Choose defaults wisely.

When first initializing the defaults, the user can access the register window to create new registers by placing the cursor in any data entry field, click the middle mouse button, and select Edit Registers. For more information about creating registers, see “Defining Registers” on page 6-2.

While working in a model, defaults for a type of network element (node, lump, path, etc.) can be set based on an object or link on the network by right clicking on the network element and selecting Set as default. All subsequent elements of this type will be identical to the object used to set the defaults. Alternatively, the user can access element-specific defaults using the Model Browser (see "Setting Network Element Defaults" on page 8-5).

FIGURE 5-3. Setting Network Element Defaults



5.2.1.3 Creating Submodels

Thermal submodels are arbitrary user-specified subsets of the master model. They can only contain thermal networking objects such as nodes and conductors, and may model phenomena such as conduction, contact, radiation, and thermal capacitance. Each submodel may have associated array data (Tables), source data, control data, output logic, and user-defined logic. The user may define submodels consisting of nodes only, conductors only, or a combination of nodes and conductors. Submodels provide a means for the user to divide a complex model into workable units or subnetworks. Examples of how the user may divide a master model into thermal submodels would be dividing by compartments, by subsystems, or by configurations. An example of dividing by configuration would be creating two separate submodels for an antenna or other deployable mechanism: one in the stowed position, and a second submodel representing the deployed position. The user could then swap out* the submodels when the mechanism is deployed during a transient.

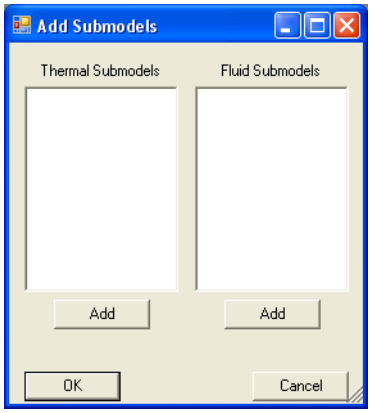
Fluid submodels are not as arbitrarily defined as are thermal submodels, because fluid submodels cannot exchange working fluid with each other: Each fluid submodel contains a single working fluid description, although the fluid may contain multiple constituents (mixture). Thus, a model of car would use one fluid submodel for the engine coolant system, one for the oil system, another for the air conditioning system, etc. Fluid submodels can only be connected to each other through heat transfer to a distinct thermal submodel.

* This capability is effected using BUILD commands. Often, a single "BUILD ALL" command is used to activate all thermal submodels at once, but subsets of the master model can be deactivated and reactivated as needed during a single run. See the SINDA/FLUINT User's Manual for more details and examples, and "Dynamically Changing the Build Configuration" on page 9-12.

A major distinction must be made between the two types of submodels. Fluid submodels are specialized to thermohydraulic modeling and have little capability for handling conduction and no capability for handling radiation. Thermal submodels, in turn, are specialized for conduction and radiation, have no capability for calculating flow resistances or pressure surges. The scope of the fluid submodel is the physics *within* the fluid container, while the scope of the thermal submodel is the physics *outside of and perhaps including* the fluid container walls.

How to add Submodels: When a new model is started, Sinaps will walk the user through initial steps including creating submodels. The initial submodel dialog shown in Figure 5-4 will appear. To add a submodel, click the Add button under the appropriate type of submodel (thermal or fluid). Provide a unique alphanumeric submodel name, underscore character '_' is allowed. Submodel names cannot exceed 32 characters. Note that long submodel names will result in larger network icons for lumps and nodes.

FIGURE 5-4. Initial Add Submodel Form

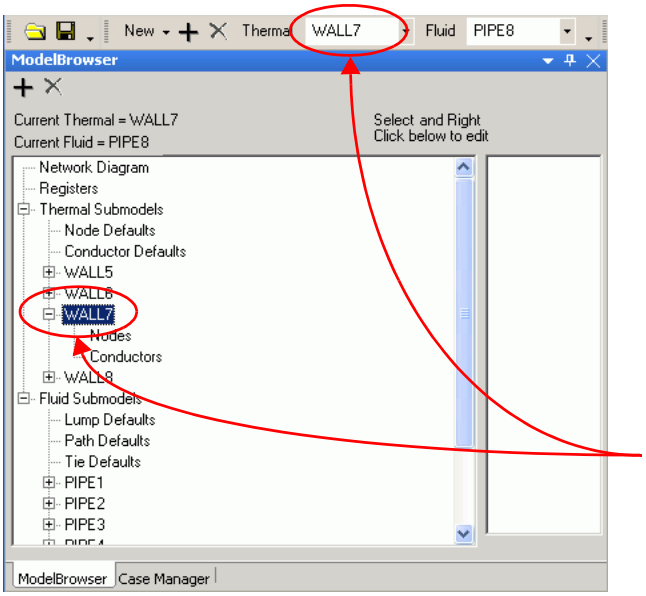


To add a submodel to an existing model, simply click the Add icon + in the Model toolbar or using the main Sinaps menu option Model>Add. A menu will appear requiring the user to specify the type of submodel to be added, thermal or fluid. After the submodel has been created it will appear in the Model Browser. Alternatively a submodel can be added through the Model Browser by clicking the plus (+) icon located in the upper left corner.

How to Delete Submodels: To delete a submodel, simply select the name of the submodel in the Model Browser tree and click on the delete icon ×. *Caution:* Once a model has been deleted, this action cannot be undone other than exiting the model without saving (and potentially losing any unsaved work).

How to Change the Current Submodel: When a node, lump, or macro is added, it will automatically be added to the *current* submodel of the appropriate type (thermal or fluid). The current thermal and fluid submodels are designated in the model toolbar. To changed the current submodel designation, either use the current submodel pull down in the toolbar or using the model browser highlight the submodel name and select the pull down menu option Model>Rename (Figure 5-5).

FIGURE 5-5. Changing the Default (Current) Submodel



Activating Submodels: At least one submodel must be active in the master model. Often the model will consist of several submodels. Submodels can be activated using the "build" options in the Case Manager (see "Case Data - Build" on page 9-10). The ***build configuration*** refers to the set of submodels currently active in a submodel during a run. A build configuration must be defined for thermal models, and a second configuration ("buildf") must be defined for currently active fluid submodels. The user can change these configurations during a run through user logic. For more information, see "Dynamically Changing the Build Configuration" on page 9-12.

5.3 Thermal Modeling Elements

This section will provides information on creating and modifying thermal modeling elements; nodes and conductors. For in-depth information on thermal elements, please see the SINDA/FLUINT User's Manual.

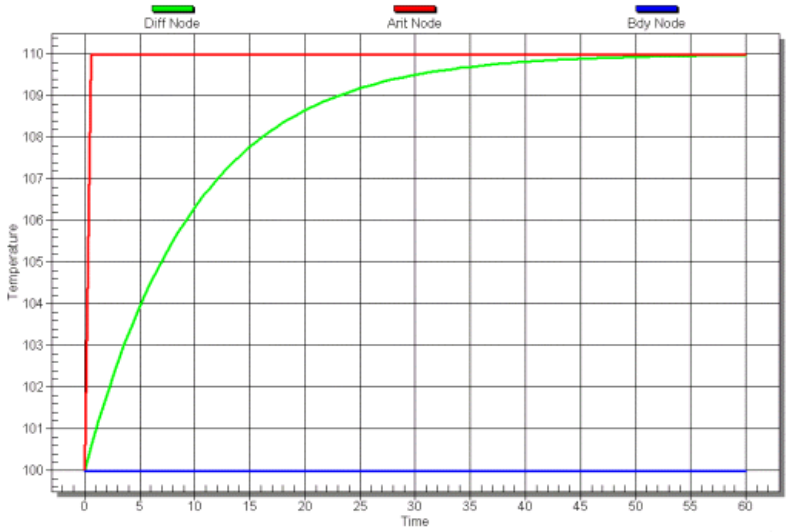
5.3.1 Thermal Nodes

Sinaps provides for the creation of three types of thermal nodes: diffusion, arithmetic, and boundary. Diffusion nodes have finite thermal mass or capacitance, and thus store and release energy. Arithmetic nodes have zero capacitance. That is, arithmetic nodes are continually brought into a steady state heat balance according to the power source applied and the heat exchange with neighboring nodes. Boundary nodes have infinite capacitance and may not receive an impressed heat source. The temperature of a boundary node is not changed during the simulation by the program, although the user can modify its value.

To illustrate the difference between the three node types, Figure 5-6 shows the transient response of the three types when a 10 watt heat source is applied. Notice the arithmetic node (which has no mass or *capacitance*) responds instan-

taneously, the boundary node does not respond at all (since it has infinite capacitance), and the diffusion node ramps up in temperature based on the thermal capacitance (m*Cp) of the node.

FIGURE 5-6. Transient Response of Various Node Types



Although the diffusion, arithmetic, and boundary nodes are the most commonly used node types, other types can be created. Heater nodes may be used to simulate heaters during a steady state solution or for the purpose of sizing a heater in a transient solution. They are the same as boundary nodes except they report the amount of heat removed or added to maintain the given temperature. This energy is calculated using the subroutine HNQCAL.

Plain diffusion nodes (material: “NONE”) are defined by a single capacitance value that is defined by a constant or single expression. Nodes with a temperature-varying capacitance can also be created. These are referred to as SIV nodes in SINDA/FLUINT: nodes whose capacitance variation may be specified as a table of Cp versus temperature, which is interpolated by SINDA/FLUINT during a transient simulation. Choosing “Temperature Varying (SIV)” enables this option, which requires an input array (containing values of T_1, Cp_1, T_2, Cp_2 ...) and a multiplying factor (usually nodal volume times density).

A more convenient way to define a temperature-varying node is to assign a material property to the “Diffusion” type, in which case the software will automatically create an appropriate SIV node given the nodal volume. “Thermophysical Property Manager” on page 5-142 provides more information on defining material properties.

Other node types (such as DIV and BIV) are also available, although seldom used. Please refer to the SINDA/FLUINT User’s Manual for more information on these node types and their required text input formats.

5.3.1.1 To create a Thermal Node

Thermal nodes may be added to the network diagram by clicking a node type in the thermal toolbar and then clicking on the network to place the node or by dragging the appropriate icon from the Tool Pallet to the diagram. Clicking on the network diagram will continue to add nodes to the diagram until either the right mouse button is selected to end *input mode* or the *select icon* in the Tool Pallet is selected and highlighted, which indicates the user is now in the

select mode. All nodes will be created based on the node defaults. To change any node parameters, the node must be edited after it has been created.

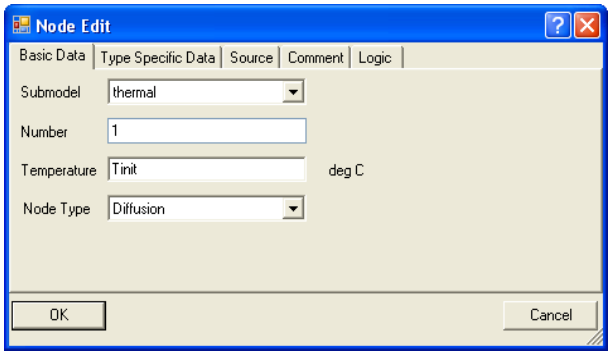
Nodes will be automatically added to the current default thermal submodel. To change this designation, choose the desired submodel in the Model Browser or in the pull-down toolbar (Figure 5-5).

5.3.1.2 Editing a Thermal Node

After a node has been created and placed in the network, it may be edited by the user. Sinaps provides two methods to edit an object: the Object Properties panel and pop-up edit forms. To access the Object Properties, click on the node in the network diagram to highlight it. The Object Properties panel will automatically be updated to show all data relevant to the selected node. (If the Object Properties panel has been closed, use View>Properties from the main workspace pull-down to re-open it.) To edit multiple nodes, hold the shift or Ctrl key down while selecting the nodes to be edited and then right click on a node to bring up the object menu and select Thermal>Edit Nodes.

To access the edit form shown in Figure 5-7, either double-click on the node to bring up the edit form directly, or select the node to be edited and click the right mouse button to display the object edit menu. Select Edit Node to bring up the edit window. The Node Edit window is a tabbed form providing access to various parameters and features. Since all node types do not require the same input, some of the data fields may be gray indicating they are not applicable for the current node type being edited. All data fields are designed to accept inputs as numeric values, registers, or expressions. For more information on expressions see “Parameterizing a Model” on page 6-1.

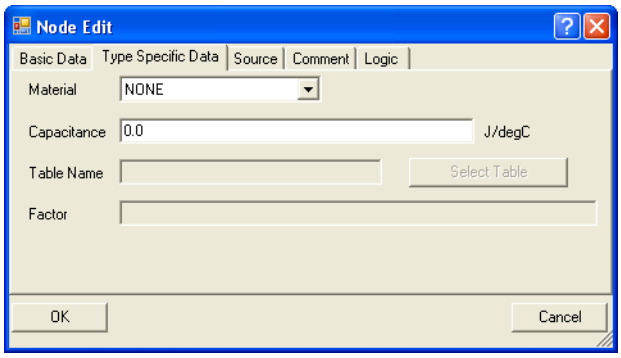
FIGURE 5-7. Node Edit Form



Basic Node Data: The tab labeled Basic Data contains data fields for defining the submodel in which the node belongs, the identification number assigned by the user (this must be an integer of up to eight digits), the temperature, and a pull-down menu for selecting the type of node. The submodel will default to the current thermal submodel, which is displayed in the submodel toolbar. The temperature field defines an initial condition for arithmetic and diffusion nodes. For boundary and heater nodes, the temperature defined will remain constant during the solution unless changed by the user in logic or via expressions.

Type Specific Node Data: The Type Specific Data tab provides access to node type-specific fields. For diffusion nodes, a material may be selected. The pull down material menu lists all materials defined in the Thermophysical Properties database. If a material is selected, the density and the specific heat values will be applied automatically and the capacitance input field becomes an input field for volume. If the user has defined the material as temperature-varying

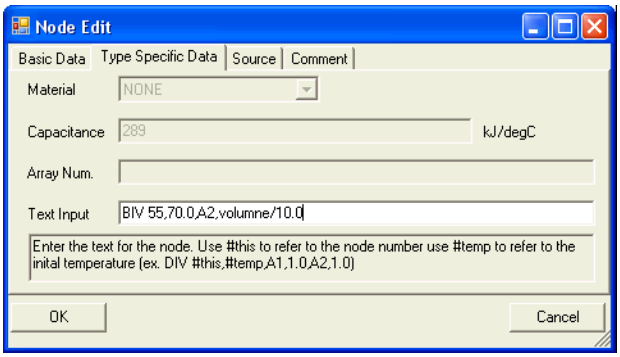
in the Thermophysical Properties database, Sinaps will recognize this dependency and will automatically create the appropriate inputs.



If the node type is selected to be Temperature Varying (SIV), the type specific data will include a Table Name field to identify the user-defined XY Table containing the temperature-varying data. Clicking on the Select Table button provides access to the table manager for selecting or creating a table (see “User-defined Data Tables” on page 5-121.) The Factor field is a user-defined value or expression that will be multiplied by the interpolated table data to define a capacitance. For example, the user could create a table of density multiplied by the specific heat and then use the factor field to define the volume, or the table may contain just specific heat and the factor field would then contain the nodal mass.

> Note: Use of the Material Property database under the Diffusion option is recommended instead of the direct SIV option, which is antiquated. The Material Property database method is both easier to use and to change, and is more self-documenting.

If the node type is selected as Other, the type-specific data will provide a field for Text Input. In this case, the expected text input is the standard ASCII format defined in the SINDA/FLUINT User’s Manual for the desired node type. Below is an example for a BIV node (one of the “other” type options) input using a time- and temperature-varying capacitance.



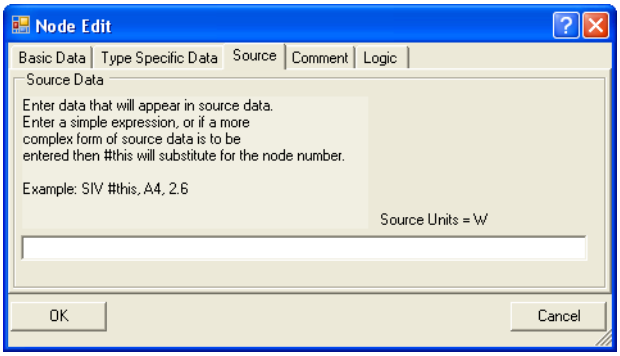

```
Basic ASCII format: BIV N#,Ti,AB,F
    where:
N# . . . . . Node ID number (integer)
Ti . . . . . Initial temperature of the node (floating point or expression)
```

AB. . . . . Reference to a bivariate array of temperature and time vs. capacitance
F. . . . . Multiplying factor (floating point data value or expression)

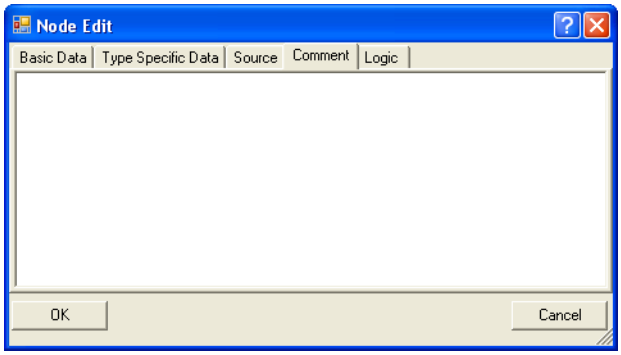
Node Source Data: The Source tab provides a means of inputting a heat source directly onto a node. The source may be defined as a numeric value, a register, or an expression. Any source data defined on this form will be output in the HEADER SOURCE DATA block. Note that when using complex forms of source data or #this referencing, the full source data line must be provided as shown in the examples below.

```
Simple Forms:
POWER*MYFACT                $ example of simple definition using registers or expressions
(TIMEN>1.0)? 50.0 : 0.0     $ example with conditional statement

Complex Forms:
#THIS, HCONV*(T#THIS-T#999) $ example of expression input with #THIS referencing
SIV 8,A4,1.0                $ example of complex SIV input
PER #this,A1,A5,1.0,.75,1.0 $ example of cyclic interpolation of time varying heat rates
```

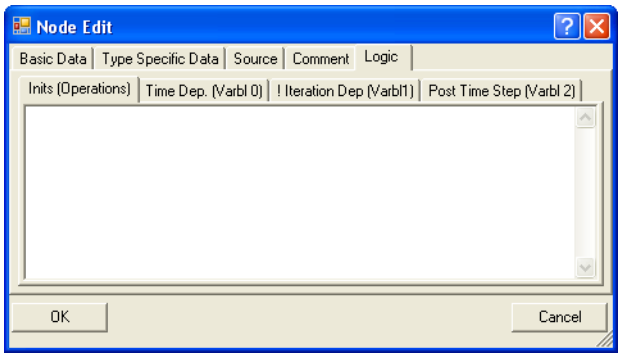
Alternatively source data can be defined through the Advanced tab of the Case Manager. Note that if multiple sources are defined for a node, they are cumulative and do not replace each other.



Node Comment: The Comment tab provides a location for the user to add a comment or description to assist with model self-documentation. The comment will be included as an inline comment in the text-based input deck.



Node Logic: The logic tab provides the ability to define network base user logic. The various tabs provide locations for an arbitrary listing of Fortran-based instructions that are associated with each node, conductor, etc. For more information see “Network Element Logic” on page 5-137.



5.3.1.3 Cloning Nodes

Clones are copies of a node that may be used purely for visual clarity: they do not affect answers. For example, a node with many conductors and ties attached to it can be cloned into two or more copies on the screen, each with its own set of connections. When the model is sent to be solved, only one node will exist in the model itself. (To verify this, try renumbering a clone: all other copies of that node will also be renumbered simultaneously.) When post-processed or colored by results, all clones will have the same color.

To create a clone, click the right-hand mouse button while hovering over the node to be cloned, and select clone from the pop-up list.

To delete a clone, delete it as if it were any other node: the other instances will not be deleted. In other words, deletion applies only to the selected clone, and not to all instances (copies) of that object.

> Tip: A common use of a clone is to make a single node appear on more than one layer simultaneously. In other words, each clone can be assigned to a distinct layer. Similarly, one clone of a node might be contained in a collection, while the other is not (or is in a

5.3.1.4 Setting Node Defaults

Node defaults can be set using the Model Browser or the Model Defaults dialog. For more information, see “Setting Network Element Defaults” on page 5-6. Refer also to Figure 5-5 for a demonstration of how to change the default thermal submodel.

5.3.2 Thermal Conductors

Conductors are network elements that describe how energy is transported between two nodes; conductors are heat transfer paths. Two basic classifications of conductors are available to the user: linear and radiation. ***Linear conductors*** are used to describe heat transfer through conduction or convection (or many other linear phenomena). ***Radiation conductors*** are used to describe radiative heat transfer as a function of temperature to the fourth power:

Linear Conductor: $HR = G \bullet (T_i - T_j)$

Radiation Conductor: $HR = G \bullet (T_i^4 - T_j^4)$ where $G = \sigma \bullet \mathcal{F} \bullet A$

Radiation conductors are a function of the Stefan-Boltzmann constant (σ). This parameter is available within expressions as the predefined or built-in registers sbcon or sbconsi (english and SI units respectively). All radiation conductors in a model get multiplied by the control constant SIGMA, which defaults to unity. SIGMA therefore represents a convenient location for containing the Stefan-Boltzmann constant.

> Note: A common mistake is to either include the Stefan-Boltzmann constant twice (both in SIGMA and in the G value of the radiation conductor), or to forget to include it (i.e., let SIGMA default to 1.0 and neglect to include σ in the conductance value). Either way, the radiation conductance will be incorrect by many orders of magnitude, and the software is often unable to trap such mistakes given that radiation conductances do typically vary by a significant amount. Therefore, extra diligence is required by the user to prevent this mistake.

Several types of linear conductors are available to the user. In addition to the standard linear conductor there are temperature-varying, time-varying, periodic, and bimetallic, just to name a few. The temperature-varying (SIV) is the most common of the "other" options. For all types other than the standard and the SIV, the user must input the conductor information as text format based on the standard ASCII text format defined in the SINDA/FLUINT User's Manual. For linear SIV conductors intended to represent a temperature-varying conductivity, a more convenient method is to define a material property, as will be explained below.

Both linear and radiation conductors can optionally be defined to be ***one-way conductors***, which are displayed as a conductor with an arrow head on one end. A one-way conductor is ***not*** a diode. Rather, it defines a direction for advection heat transfer, or motion of solid mass (though there are other uses as well). The downstream node (the node at the arrow end of the conductor) sees the effect of the conductor as if it were a normal two-way conductor, whether energy is leaving that node or arriving at it through the conductor (i.e., independent of the direction of temperature gradient). Conversely, the upstream node does *not* see the downstream node nor any energy from it, as if the conductor did not exist. Using one-way conductors can disrupt the system energy balance required for steady state convergence. Refer to the SINDA/FLUINT User's Manual for more details on this specialized modeling tool.

5.3.2.1 To Create a Thermal Conductor

Two methods are available for creating a conductor in the model network:

1. click on the conductor type icon on the thermal toolbar or on the Tool Pallet to select it. If using the Tool Pallet, grip points will appear on the icon highlighting it. Next, hover over any node on the network until the cursor changes appearance to a hand. Holding the left mouse button down, simply draw a line from the first node to the second node to create a conductor. As long as the conductor icon is highlighted in the tool bar, multiple conductors can be created in this fashion.
2. using the Tool Pallet or thermal toolbar, click the type of conductor (linear or radiation) to set the default and then draw the link between two nodes on the network. When placing the cursor over the node, located it near the perimeter of the icon so the hand icon appears ( ) to enable the draw mode. Alternatively hover the cursor over the first node, press and hold the shift key while left clicking on the object. Hold the shift key as the cursor is pulled away from the node and the link is drawn to the second node.

All conductors created will be defined based on the conductor defaults. The user can modify the conductor parameters by editing the conductor.

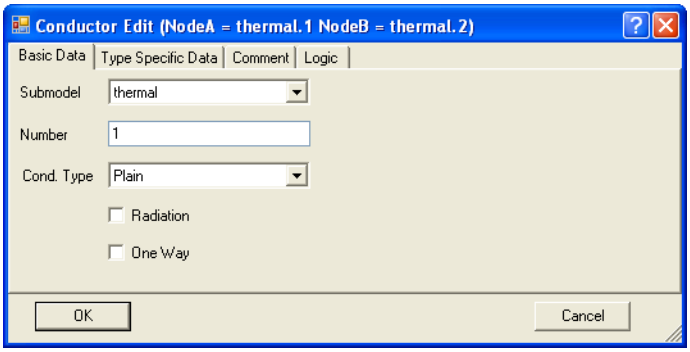
5.3.2.2 Editing Conductors

A conductor can be edited either using the Object Properties window or the Conductor Edit form. The Conductor Edit form shown in Figure 5-8 can be accessed by double-clicking on the conductor. Alternatively, clicking on a conductor will highlight it (meaning its grip points will become visible). Once selected, the conductor data is displayed in the Object Properties window and can be edited directly. (If the Object Properties window has been closed, it can be re-opened using View>Properties from the main workspace pull down menu.)

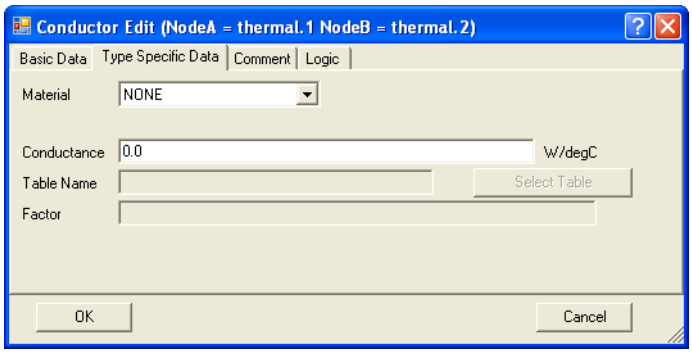
To edit multiple conductors, hold the shift or Ctrl key down while selecting the conductors to be edited and then right click on a conductor to bring up the object menu and select Thermal>Edit Conductors.

Basic Conductor Data: The Conductor Edit window is a tabbed form providing access to various input options. The Basic Data tab provides access for selecting the submodel in which the conductor will reside, providing an ID number (integer identifier of up to eight digits in length), and selecting the type of conductor: linear, temperature-varying (SIV) or other (DIV, SIM, BIV etc.). Check box options are provided for specifying a radiation or one-way conductors. By default, the conductor will be created in the current submodel displayed in the model toolbar.

FIGURE 5-8. Conductor Edit Form



Type Specific Conductor Data: The Type Specific Data tab provides access to the various conductor data fields. The fields provided on the form will vary depending on the type of conductor. The conductor field is designed to accept inputs as numeric values, registers, or expressions. For more information on expressions see “Parameterizing a Model” on page 6-1. This form also provides access to the list a materials defined in the Thermophysical Property database as an alternative to inputting the full conductance directly. If a material is selected for a linear conductor, the conductance field name will change and the input becomes Area/Length. The conductivity will be automatically applied based on the material database.

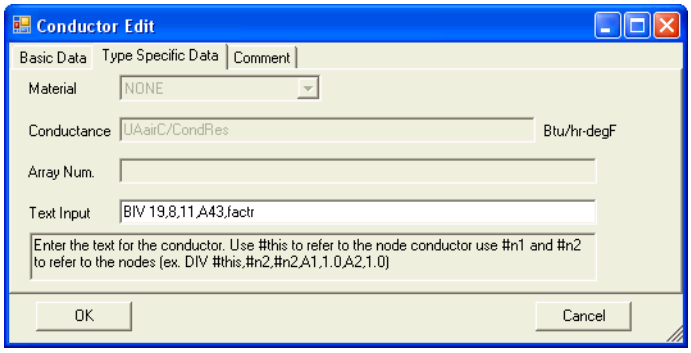


If the conductor type is selected as Temperature Varying (SIV), the Table Name field should reference the XY table containing the temperature vs. conductivity data. Clicking on the Select Table button provides access to the table manager for selecting or creating a table (see “User-defined Data Tables” on page 5-121.) The factor field can be used to define a multiplier for the conductivity such as area/length.

> Note: Use of the Material Property database under “type specific data” is the recommended option instead of the direct SIV option, which is antiquated. In this situation the conductor type is left as “Plain” and the material is defined with temperature varying properties through the Thermophysical Property Database Manager. This property database method is both easier to use and to change, and is more self-documenting than the use of SIV conductors.

If type Other is selected for the conductor, the user must specify the conductor input in the Text Input field using the standard ASCII input format for the specific type of conductor to be created, BIV, SIM DIV, TVS, PER, etc. For all available conductor types, please reference the SINDA/FLUINT User’s Manuals section 2.14.1.

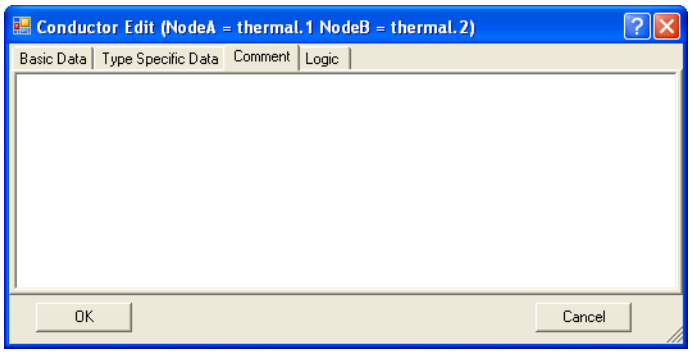
Example of a time- and temperature-varying conductor (type BIV using the “other” type option):



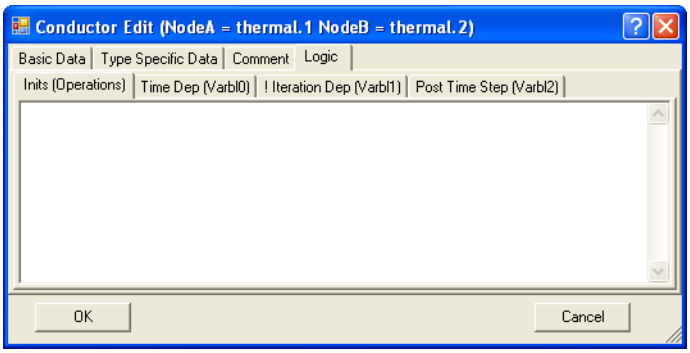
Basic Format: BIV G#,NA,NB,APB,F $(basic format)
where:
G# . . . . . Conductor ID number (integer)
NA,NB . . . . . ID numbers of the pair of nodes to which this conductor is connected. To pass along the

. . . . . attache nodes, use #n1 and #n2 (note the indirect references #nodea and #nodeb will not
. . . . . work in this field.
APB. . . . . Reference to a bivariate array (of temperature and time vs. conductance points)
of the form: An or smn.An, where n is the array number
F. . . . . Multiplying factor input (floating point data value or expression)

Conductor Comment: The Comment tab provides a location for the user to add a comment or description to assist with model self-documentation. The comment will be included as an inline comment in the text-based input deck.



Conductor Logic: The logic tab provides the ability to define network base user logic. The various tabs provide locations for an arbitrary listing of Fortran-based instructions that are associated with each node, conductor, etc. For more information see “Network Element Logic” on page 5-137.



5.3.2.3 Setting Conductor Defaults

Conductor defaults can be set using the Model Browser or the Model Defaults dialog. For more information, see “Setting Network Element Defaults” on page 5-6. Refer also to Figure 5-5 for a demonstration of how to change the default thermal submodel.

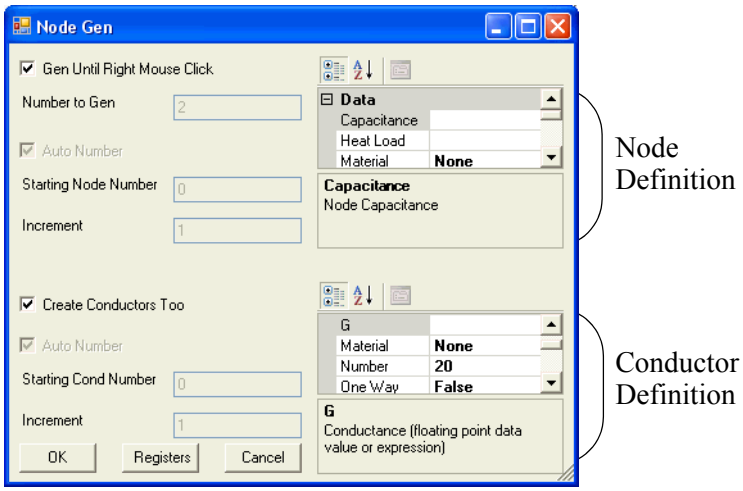
5.3.3 Generating Multiple Node and Conductors

Sinaps provides a node generation option to create multiple nodes and conductors. To invoke this option, select the “Generate Nodes and Conductor” icon in the toolbar, or select the Node Gen option from the Tool Pallet to bring of the

dialog shown in Figure 5-9. This form allows the user to specify parameters for the nodes and conductors to be generated on the right side of the window. The parameter list is scrollable and provides access to all input parameters for both the nodes and conductors. The default is to generate diffusion nodes and linear conductors. Upon exiting the form, nodes and conductors between them will be generated every time the user clicks on the network, until the right mouse button is clicked. If the Create Conductors Too box is unchecked only, nodes will be created.

Alternatively the user can define how many nodes are to be created by unchecking the Gen Until Right Mouse Click button. Upon exiting the form, the user must place each node on the diagram. If the user opts to not use autonumbering, the starting number and increment must be defined for the node IDs to be generated.

FIGURE 5-9. Node and Conductor Generation Dialog



5.4 Flow Modeling Elements

Fluid submodels are comprised of lumps, paths, heat transfer ties, interfaces (ifaces), fluid to fluid heat transfer ties (fties), and macros. Every fluid submodel also has one or more associated working fluid species. This section provides information on defining fluid network elements and associated working fluids.

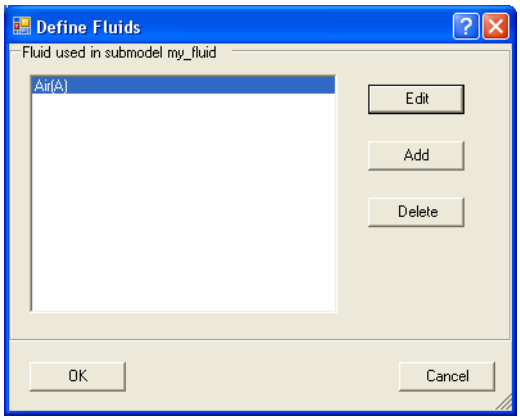
5.4.1 Working Fluids

The user must define at least one working fluid for each fluid submodel. Fluid cannot be exchanged between fluid submodels, although multiple submodels can share the same working fluid description. The *fluid description* which may either be one of 21 built-in fluid descriptions (air, water,* ammonia, etc.) or a user-provided description in the form of

* The built-in water description is highly approximate. More complete water properties are available at www.crtech.com, along with many other fluids such as cryogens, refrigerants, and fuels.

an INCLUDE or INSERT file containing an FPROP DATA block. Submodels may either use a single fluid or a mixture of two or more fluids.*

For each fluid submodel, the associated fluid definition dialog can be accessed though the Model Browser as shown in Figure 5-10. Using the Model Browser, expand the Fluid Submodels tree and select Fluids for the appropriate fluid submodel. A list of the current working fluid species will be visible in the right-hand window. By default, the fluid is set to the built-in description for air and assigned species designation "A." This default can be changed simply by locating the cursor over the fluid identifier and clicking the right mouse button to bring up the Edit Fluids option. Clicking the edit option will open the Define Fluids dialog window.



The Define Fluids dialog allows the user to select either a built-in (library) fluid using the pull-down list or a user-defined fluid. If a user-defined fluid is selected, click the Fluid Includes button to bring up a listing of available INCLUDE† files. If the model is new, this list will be empty. However clicking on the "..." on at the right end of a line will bring up a browser window allowing the user to locate the desired INCLUDE file. Sinaps will automatically detect the fluid number in the file and place it in the fluid ID field.

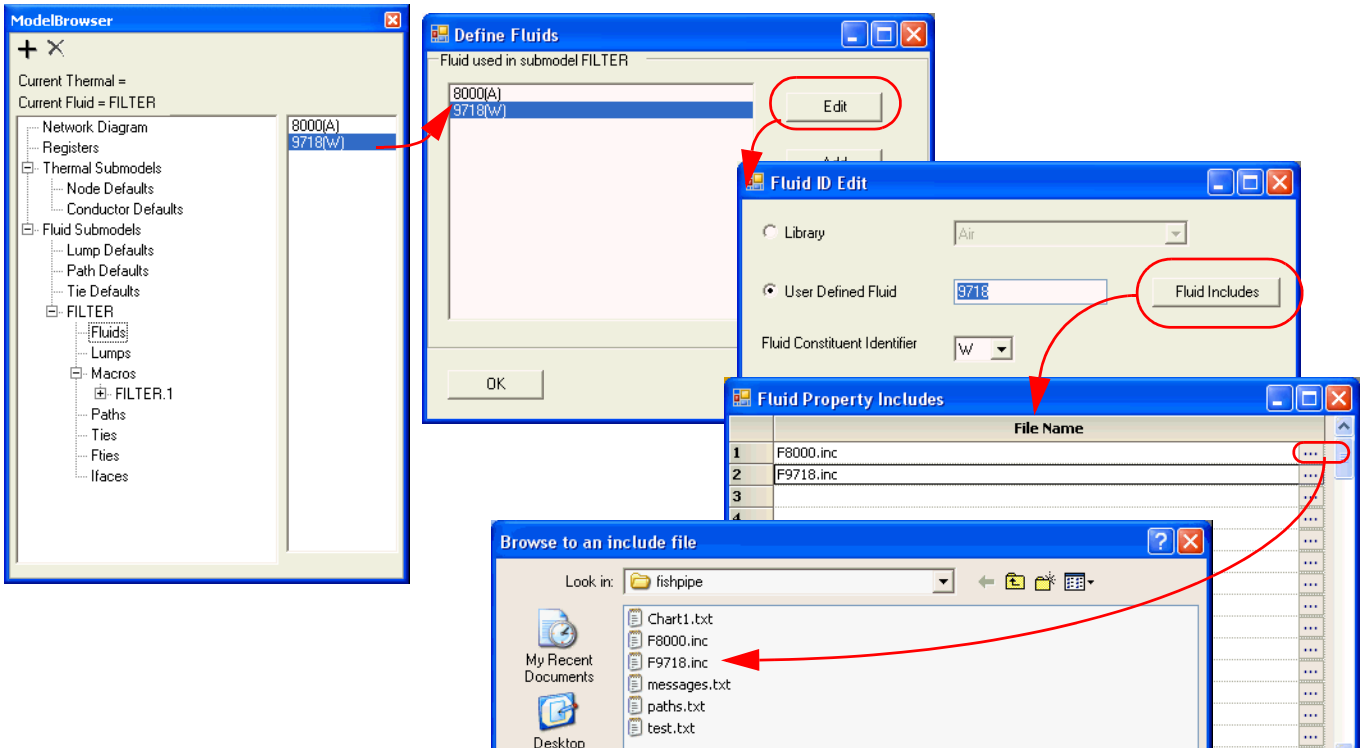
When adding a second fluid submodel which uses the same fluid as a previously created fluid submodel, when the Fluid Property Includes dialog appears, select the row with the include file to be used in the new submodel and click OK. If you browse to the same file it will duplicate entries in the Fluid Property Includes dialog, causing duplicate insert statements and eventual preprocessor errors.

> Note: The Property Fluid Includes dialog is a global level window, i.e. it is applicable for all fluid submodels. Consequently it will list all include files used in the master model. For example if you have 3 fluid submodels using 5 user-defined fluid descriptions, the dialog will list 5 include files. Two separate fluid submodels can point to the same fluid include file if they use the same fluid.

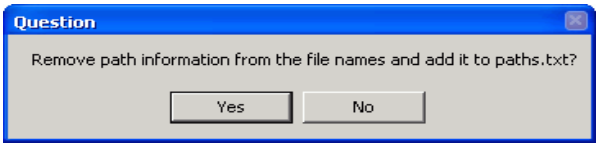
* Working fluid mixtures can be a complex subject with many modeling options available. Refer to the SINDA/FLUINT User's Manual for details.

† The INSERT command is used in preference to the INCLUDE command. The name "INCLUDE file" is a historical misnomer.

FIGURE 5-10. Specifying Fluids



After browsing for an INCLUDE file, the user will be presented with the following dialog:



Selecting “Yes” will cause Sinaps to strip off file path designations from the name, and it will write them to a local “paths.txt” file. For example, it will replace “C:\myfile\myfluid.inc” with “myfluid.inc” and place “C:\myfile” in the paths.txt file as a valid INCLUDE file directory location to be searched when running the model. The purpose of this recommended option is to encourage the centralization of fluid properties, and to make Sinaps models more transportable to third parties (who will inevitably use a different directory structure). To learn more about the paths.txt file and it’s full usage, please see section 2.5.2 of the SINDA/FLUINT User’s Manual.

The fluid constituent identifier must be specified if the fluid is to be used in a mixture. This identifier is a single letter (A through Z) designated by the user. If only one fluid exists in the submodel, this can be left at the default value of “A.” For multiple constituents, additional species can be “added” using the Define Fluids form. Each fluid must have a unique fluid constituent ID assigned to it. Within each submodel, the same constituent must be consistently referenced

using the letter designator. For this reason, choosing a mnemonic is recommended: “W” for water, “A” for air, etc. These identifiers are submodel-specific, so “A” may represent air in one submodel and ammonia in a second submodel.

> Note: If the fluid identifier (A through Z) is changed after assigning mixture fractions to lumps (see “Setting Initial Mass Fractions for Mixtures” on page 5-35), the change will not automatically propagate to the lump data. The user must edit lumps as necessary to redefine fluid IDs assigned with the initial mixture fractions.

User-defined Fluids: Four types of user-defined fluids are available. These fluid descriptions can either be a pre-defined INCLUDE file or the user can create his/her own fluid description. For extensive instructions on defining fluids, please see Section 3.21 of the SINDA/FLUINT User’s Manual. The fluid types are identified by using four-digit fluid numbers, where the first digit identifies the type:

1) A perfect gas (that cannot condense): “***8000 Series***.”

   Both the specific heat and transport properties may vary with temperature. In other words, the gas does not have to be calorically perfect. The fluid identifiers for these fluids can range from 8000 to 8999. 8000 series fluids may be used as constituents within mixtures. They may also be used as solutes in mixtures.

2) A simple liquid (that cannot boil): “***9000 Series***.”

   The liquid is assumed to be incompressible, but density as well as transport properties can vary with temperature. The fluid identifiers for these fluids can range from 9000 to 9999. 9000 series fluids may be used as constituents within mixtures. It may optionally contain a saturation curve such that the validity of its “never boil” assumption can be tested.

3) A simplified description of a two-phase fluid: “***7000 Series***.”

   This option is somewhat like a combination of the above two fluids, with saturation dome and two-phase properties also required. This option is intended to allow a limited description of a two-phase fluid to be quickly added as an alternate fluid using commonly available properties. The fluid identifiers for these fluids can range from 7000 to 7999.

4) An advanced (complete) description of a two-phase fluid or real gas: “***6000 Series.***”

   The user may replace some or all of the standard library property routines with alternates; in other words, the user may input a functional description of a fluid over a very wide range of properties, including supercritical if desired. The fluid identifiers for these fluids can range from 6000 to 6999. This fluid description can encompass thermodynamically compressible liquids as well as real gases.

5.4.2 Defining Fluid Mixtures

Each fluid submodel may contain up to 26 individual fluids, each of which is a species that will be tracked as part of the working fluid mixture. Such mixture are currently limited to zero or one condensible/volatile species (library fluid, or 7000 or 6000 series fluid), and zero or more perfect gases (8000 series), real gases (6000 series NEVERLIQ fluids),

and simple incompressible liquids (9000 series). One or more of the perfect gases (8000 series fluids) can be soluble in one or more of the liquids.

When more than one fluid is defined in the Define Fluids dialog window, the user must initialize the appropriate lumps in the network for mixture gas mass fraction and liquid mass fraction. If multiple fluids have been defined, a Mixture tab will be present on the Lump Edit dialog when creating lumps. This tab allows the user to set initial mass fractions of each species. For more information see "Setting Initial Mass Fractions for Mixtures" on page 5-35.

5.4.2.1 Mixture Rules

Each fluid is usually assumed to behave ideally and independently when mixed with others. In other words, when real or perfect gases mix, a partial pressure approach is used. Transport properties of gases are calculated on the basis of a weighted average of the partial pressure of each constituent.

Note that the properties of a liquid mixture are also dependent on whether or not the liquids have been declared to be immiscible (not capable of being mix). Liquids that are immiscible are treated separately, using their individual species' properties. The properties of miscible liquids are determined using a variety of correlations or other estimation methods. For liquid mixtures consisting of some miscible and some immiscible liquids, the final bulk properties are calculated using a volumetric average. For example, consider 3 liquids A, B, and C, of which A and B are miscible and C is immiscible. The density of the mixture is calculated by dividing the total liquid mass by the volume of fluid C plus the volume of the A-B mixture. Other properties (surface tension, viscosity, conductivity, compliance and sound speed) follow analogous rules.

When a condensible/volatile species is present, the presence of other substances is assumed to have no effect on its boiling point (other than the effect of partial pressures in the gaseous phase).

Mixture Data (Solubility): By default, gases are assumed to have zero solubility into any liquids with which they are mixed. The user can choose to model equilibrium and nonequilibrium dissolution and evolution of any noncondensible gas (8000 series user-defined fluid) into any fluid with a liquid phase. To define the solubility of gases, ***Mixture Data*** must be used to defined the dissolution properties. The Mixture Data form is accessible through the Advanced tab of the Case Manager under Fluid Property Data>Mixture Data. This form, like other data input forms, will provide a description of the required input for each data field when the cursor is held over any data field (viewable in the lower portion of the form). For details of the required inputs, and more about using and defining mixtures, please see section 3.19 of the SINDA/FLUINT User's Manual.

5.4.2.2 Mixture Output

Two standard tabular routines are available to present constituent concentration information within each lump: CNSTAB and LMCTAB. These subroutines must be called from an output block via user-defined logic. LMCTAB is intended for models with less than 8 constituents. It tabulates constituent mass fractions for each lump. Whereas LMCTAB tabulates by lump, CNSTAB tabulates by constituent. It provides more detailed information about the amounts of all constituents throughout the model.

5.4.3 Fluid Lumps

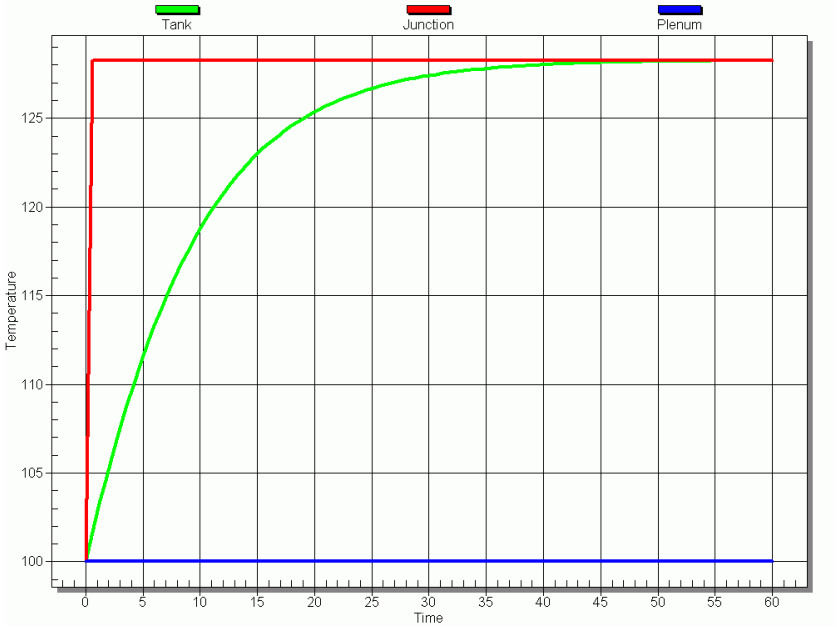
Sinaps provide for the creation of three types of fluid lumps: tanks, junctions, and plena. ***Tanks*** are finite control volumes that may exchange energy with the environment and may grow or shrink. Because they are governed by differential equations for mass and energy in time, tanks cannot change instantaneously. ***Junctions*** can be conceptualized as

zero volume tanks that may also exchange energy with the environment. Junctions react instantaneously because they are governed by algebraic (time-independent constraint) equations for mass and energy. They may be used to model tees, small tanks, or points where flow information is needed. ***Plena*** (plural of "***plenum***") can be conceptualized as tanks with infinite volume. They represent boundary conditions for modeling open systems; plena are sources or sinks of fluid.

Each lump has a single characteristic thermodynamic state, and therefore inherently presumes perfect mixing. To simulate lack of thermal equilibrium between phases when liquid and vapor or gas are both present, pairs of tanks ("twinned tanks") may be input and *phasic nonequilibrium* simulation may be invoked.

To illustrate the difference between the three lump types, Figure 5-11 shows the transient response of the three types of lumps when a 10 watt heat source is applied. Notice the junction (which has no volume) responds instantaneously, the plenum does not respond at all since it has infinite volume, and the tank ramps up in temperature based on the mass and state of the fluid contained within the volume. (The response shown is for a single-phase liquid.)

FIGURE 5-11. Transient Response of Various Lump Types



5.4.3.1 To create a fluid lump

Lumps may be added to the network diagram by selecting a lump icon from the fluid toolbar or by dragging the appropriate icon from the Tool Pallet to the diagram. In order to create several lumps of the same type using the Tool Pallet, click on the type of lump to be created in order to highlight it, and then locate the cursor in the network diagram and simply click the left mouse button to place a new lump. Clicking will continue to add lumps to the diagram until either the right mouse button is click to end *input mode* or the *select icon* in the Tool Pallet is selected and highlighted.

All lumps will be created in the current submodel as defined in the submodel toolbar, and the input parameters will be based on the lump defaults defined in the Model Defaults dialog. To change any input value from the default, the lump must be edited.

Setting lump states: All lumps have a single specific thermodynamic state. This state must be initialized by the user, but can be customized in logic blocks via calls to the CHGLMP routine (see section 7.10.1.1 of the SINDA/FLUINT User's Manual). While many properties describing the lump state are available for the user's convenience in logic blocks, only three properties are required to determine the initial lump state for single-constituent working fluids. These input properties are:

```
TL . . . . . Lump temperature
AL or XL. . . . . Void fraction (vapor volume fraction) or Quality (vapor mass fraction)
PL . . . . . Lump pressure
```

While only two properties (TL, PL) are required to completely specify a single-phase single constituent state, the pressures and temperatures are not independent for two-phase states of pure substances. Therefore, a third property, void fraction (AL, vapor volume fraction) or quality (XL, vapor mass fraction), is also needed. Since there is a potential conflict between these three properties for single-phase states (where only two are required), by default Sinaps assumes that TL and AL/XL are always correct, but that PL may be a first guess and hence will be overridden if it conflicts with TL and AL/XL. In other words, a hierarchy exists:

1. AL/XL is always used
2. TL is used
3. PL is overridden if it conflicts thermodynamically with TL and AL/XL.

The user may replace this hierarchy by using the pressure priority option, signaling that the input pressure value is to be given priority over the input temperate value, which may be overridden if it conflicts with PL and AL/XL. The exclamation point is used to designate the pressure priority option as "PL!" with the following rules applying:

1. AL/XL is always used
2. PL! is used
3. TL is overridden if it conflicts thermodynamically with PL! and AL/XL

Note: Unlike a thermal node, which can have the temperature easily reset in logic (WALL.T25=300.), the parameters which define a lump's state (temperature, quality and pressure) are interdependent. Therefore, the subroutine CHGLMP must be used to reset parameters associated with a lump's state. This routine allows the user to reset one variable and specify which variable is to be held constant, thus allowing the third variable to change as needed. CHGLMP is normally called in OPERATIONS to set the initial conditions for a tank or junction, but may also be called in FLOGIC 0 to reset a plenum's state during a solution.

```
CALL CHGLMP (SUBMOD_NAME, LUMP_NO, VARY, NEW_VALUE, HOLD)
Example:
CALL CHGLMP ('COOLER', 123, 'TL', 300.0, 'PL')
```

Use of these rules is summarized in Figure 5-12. See also "Lump Initialization Wizard:" on page 5-32.

FIGURE 5-12. Lump State Initialization (without "PL!")

Desired State	Input Quality	Input Pressure
Subcooled Liquid	XL = 0.0	PL > P_{sat}(TL) *
Saturated Liquid	XL = 0.0	PL ≤ P_{sat}(TL) **
Two-phase	0.0 > XL > 1.0	anything **
Saturated Vapor	XL = 1.0	PL ≥ P_{sat}(TL) **
Superheated Vapor	XL = 1.0	PL < P_{sat}(TL)

* P_{sat}(TL) is the saturation pressure corresponding to TL.

** PL will be replaced with P_{sat}(TL) in these cases. NOTE: 6000 series fluids permit supercritical vapor, so if XL=1.0, then use PL > PGMAX or PL < P_{crit} for saturation, else P_{crit} < PL < PGMAX for supercritical.

Duct Modeling: There are important differences between lumps and paths generated or input separately, and those generated in order to represent a continuous duct that has been subdivided (perhaps as needed to model heat transfer or side flow passages). The difference is that the code assigns no higher-level meaning to a simple, perhaps chance catenation of lumps and paths, whereas a ***duct*** is known to represent a continuous flow passage. Additional terms and methods are applied to ducts that are not applied to elements that are input separately.

Therefore, the analyst should preferentially use duct (pipes and auto generated ducts) whenever modeling an axially subdivided duct or other continuous flow passage. For more information on duct macros, including automatic generation of them by Sinaps (when it auto-detects a continuous flow passage), see "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5.

Twinning tanks: When modeling two-phase systems, each lump is assumed to be homogeneous with the two phases perfectly mixed. In the majority of two-phase models, this assumption is appropriate. If this assumption is inappropriate for a particular model, the user may override it by specifying ***twinned tanks***.

By default tanks are not twinned and the user can use the twin option on the tank edit form to twin the tank if so desired. During execution, the twinned tank will be treated as two tanks: one liquid and one vapor, both sharing the *total* volume of the tank. When tanks are twinned, a superpath, an IFACE, and a FTIE will be automatically created between the twins to control the movement of mass and energy between the vapor and liquid tanks. When the tanks are single-phase liquid or vapor (or nearly so), then the secondary tank automatically disappears temporarily and is

ignored. The user may optionally twin paths (tubes or STUBE connectors) between sets of twinned tanks to model the discrete velocities of liquid and vapor.

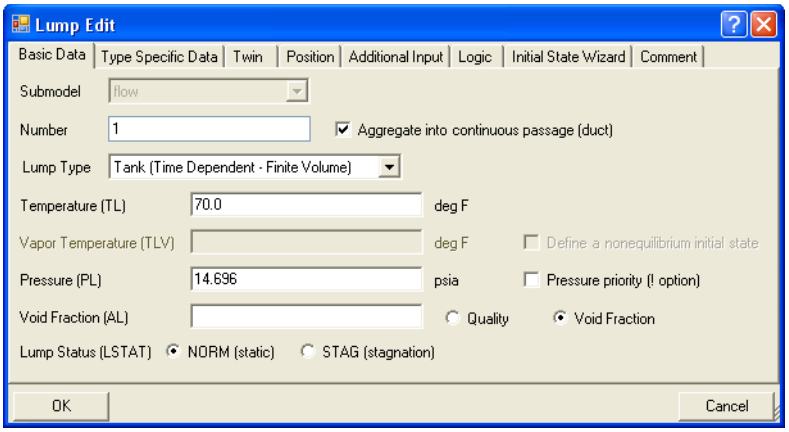
Before using twinned tanks or twinned paths, it is highly recommended that the user review the full documentation on the subject in section 3.25 of the SINDA/FLUINT User's Manual.

5.4.3.2 Editing a Lump

After a lump has been created and placed in the network, it may need to be edited. Sinaps provides two methods for editing a lump: the Object Properties task panel and the Lump Edit form. To access the Lump Edit form, either double click on the lump icon to bring up the edit form directly, or select the lump and click the right mouse button to display the object edit menu. Select Edit Lump to bring up the edit window. To edit in the Object Properties panel, click on the lump in the network diagram to highlight it. The Object Properties panel will automatically be updated to show all data relevant to the selected lump. If the Object Properties panel is not visible, it can be opened through the main workspace menu option View>Properties.

The Lump Edit window is a tabbed form providing access to the most commonly used input parameters. Since all lump types do not require the same input, some of the data fields may be gray, indicating they are not applicable for the current lump. All data fields are designed to accept inputs as numeric values, registers, or expressions. For more information on expressions see "Parameterizing a Model" on page 6-1.

Basic Lump Data: The Basic Data tab provides access to parameters common to all lumps including submodel, ID, temperature, pressure, quality, and the lump status options. The submodel is grayed out and cannot be changed from this form.*



Number - The identification number of the tank. Numbers are automatically assigned by Sinaps but can be overridden by the user.

Aggregate into continuous passage (duct) - When checked, this option flags Sinaps to include the lump as part of automatically generated ducts (see "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5). By default this

* Such a change would be unusual since it would also normally mean a change of working fluid. Also, such an action will delete any connections to that lump. Therefore, while the fluid submodel can be changed using the right-click option Move to fluid submodel, this option should be used sparingly.

option is enabled, but it can be disabled by the user if necessary. The automatic duct option can be globally disabled through the preferences dialog, see “Setting Preferences” on page 3-11. In general, *it is recommend that the option be left enabled unless the user needs to locally override the aggregation into an automatic duct*

Lump Type - Select the type of lump to create, plenum (infinite volume and pressure source), tank (volume) or junction (no volume).

Temperature (TL) - Define the initial vapor temperature for tanks and junctions. Temperature to be maintained for a plenum.

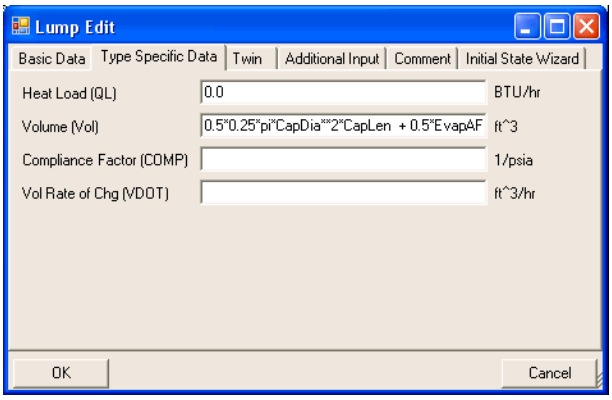
Vapor Temperature (TLV) - Define the initial temperature of the secondary (vapor/ullage) side of a twinned tank making it distinct from that of the liquid side. See “Twinning tanks:” on page 5-26. If this parameter is not specified, then the twinned tanks will be initially in the combined (equilibrium) mode. If it is specified (usually TLV>TL), then the primary and secondary tanks will be initialized separately, and the *first solution is expected to be a transient (and not STEADY)*. Use of both AL and PL! are strongly recommended when using the TLV input option. Further guidance is provided in Section 3.25.6.2 of the SINDA/FLUINT User’s Manual.

Pressure (PL) - Define the initial pressure for tanks and junctions. For plenum this will be the pressure to maintain. The check box below the pressure data field allows the user to define the pressure priority option (pressure and quality are set and temperature may be overridden as explained in “Setting lump states:” on page 5-25). By default, the box is unchecked and temperature has priority.

Quality (XL) - The initial quality of the lump. For a plenum, the quality and the temperature will define the state unless the pressure priority option is used.

Lump Status (LSTAT) - For plena, the fluid is assumed to be stagnant (LSTAT = STAG) by default, causing acceleration losses to be automatically added to most exiting paths. When a lump is designated as stagnant (meaning zero velocity), choking calculations are also applied to the out-flowing paths to include modeling of the expansion process. Also, kinetic energy is neglected at the entrance (but not exit) of suitable paths flowing out of a stagnant lump. Hence the pressure of the lump (PL) is assumed to be the stagnation (or total) pressure, instead of static. Tanks and junctions, on the other hand, are assuming to represent moving fluid, and are defaulted to LSTAT = NORM (meaning PL is the static pressure). If the tank represents a large reservoir or vessel, etc. and does not simply represent the pressure at a point along a pipeline, then the STAT option should be changed to be STAG. For more information about see section 3.15.2.5 “Inlets and Outlets” of the SINDA/FLUINT User’s Manual.

Type Specific Lump Data: The Type Specific Data tab allows access to parameters which are specific to the type of lump being created (i.e., junction, tank, or plenum).



Heat Load (QL) - Heat source to be applied to the lump. Applicable only to tanks.

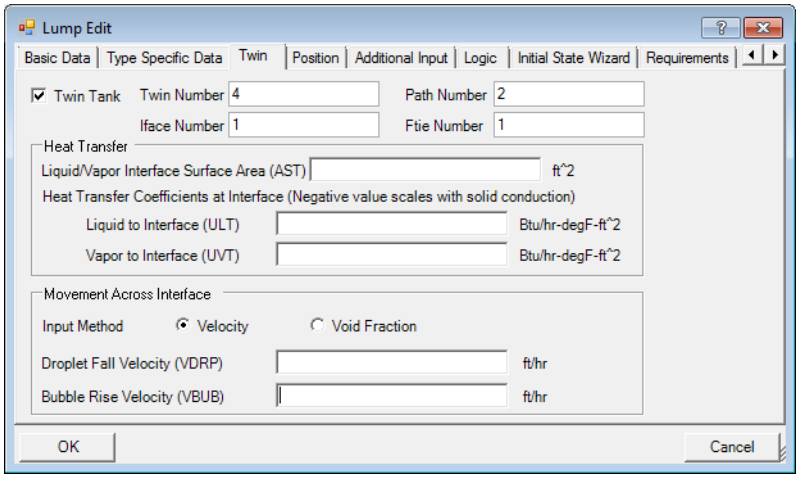
Volume (Vol) - The total volume of the tank. For tanks, the user must define a volume. For junctions, the volume is not required although the data field is present as a aid to the user. Junction volume can be used to calculate the mass it *would* contain, and its presence also facilitates switching the lump between being a junction and a tank. The volume of a junction will be ignored in almost all calculations with two exceptions, tracking system volume or mass using the SUMFLO or SUMDFLO routines, or modeling homogeneous nucleation of dissolved gases. Please refer to the SINDA/FLUINT User's Manual for more information on these exceptions.

Compliance Factor (COMP) - A tank wall compliance can be used to model gas inclusions, container flexibility, and liquid compressibility (if the liquid is otherwise incompressible). Pressures may change too quickly or severely in a *hard* (all liquid) tank since there is no place to store or release any excess fluid. Tank wall compliance factors can help reduce this problem by removing the unrealistic but useful assumption of inflexible walls. Generally *soft* (two-phase or vapor) tanks do not need any compliance.

V*ol Rate of Chg (VDOT)* - The volumetric growth rate. Applicable only to tanks.

Defining Twinned Tanks: Twins are used when the assumption of homogeneous fluid and perfect mixing between phases is not appropriate. Note the twin option is only available if the lump is defined as a tank. If a tank is to be twinned, the secondary tank initial condition can be. specified on the Basic Data tab using the TLV option (see "Basic Lump Data:" on page 5-27). For additional information on twinning tanks please see "Twinning tanks:" on page 5-26 and Section 3.25 of the SINDA/FLUINT Users' Manual, "Modeling Phasic Nonequilibrium using Twinned Tanks".

The Twin tab dialog for a tank is shown below.



Twin Tank - To activate the twin option for a tank, check the Twin Tank box.

Twin Number - The identification number of the secondary tank (vapor/ullage). Numbers are automatically assigned by Sinaps but can be overridden by the user.

Path Number - The identification number of the superpath automatically created between the twinned tanks. Numbers are automatically assigned by Sinaps but can be overridden by the user.

Iface Number - The identification number of the IFACE automatically created between the twinned tanks. Numbers are automatically assigned by Sinaps but can be overridden by the user.

Ftie Number - The identification number of the FTIE automatically created between the twinned tanks. Numbers are automatically assigned by Sinaps but can be overridden by the user.

Liquid/Vapor Interface Surface Area (AST) - This is the surface area of the interface between the two lumps. Defaults to 0.0, meaning no additional heat or mass transfer will occur across the main interface (other than that portion inherited from adjacent paths). Specify -1.0 to use a spherical bubble approximation or use a negative multiplying factor.

Liquid to Interface (ULT) - This is the heat transfer coefficient between the liquid and the interface. Defaults to -1.0, meaning essentially solid conduction. Specify a negative multiplying factor or a positive value to use directly.

Vapor to Interface (UVT) - This is the heat transfer coefficient between the vapor and the interface. Defaults to -1.0, meaning essentially solid conduction. Specify a negative multiplying factor or a positive value to use directly.

Movement Across Interface - When liquid appears in the secondary/gas/vapor side of a twinned tank, it is returned gradually to the liquid side as droplets at a velocity VDRP. VBUB similarly prescribes the bubble rise rate of vapor leaving the primary/liquid side and returning to the vapor/gas twin.

> *Input Method* - The rate of bubble rise or droplet fall can be defined using either velocities or void fraction limits for the primary and secondary tanks. The void fraction limit method is an alternative which can be used to keep the tank from getting too wet (VDRP) or too dry (VBUB).

Droplet Fall Velocity (VDRP) - The velocity of droplets in the vapor (secondary) tank towards the interface. Defaults to 3600 ft/hr (1 ft/s, 0.3048 m/s), but can be changed by the user.

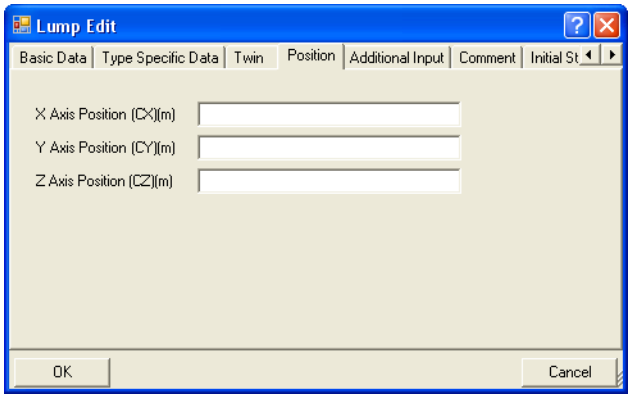
Bubble Rise Velocity (VBUB) - The velocity of bubbles in the liquid (primary) tank towards the interface. Defaults to 3600 ft/hr (1 ft/s, 0.3048 m/s), but can be changed by the user.

Vapor Side Lower Limit (VDRP) - The minimum void fraction (AL) in the secondary (vapor) tank in the presence of liquid injection or continuous condensation. The void fraction should be a positive value between 0.001 and 1.0.

Liquid Side Upper Limit (VBUB) - The maximum void fraction (AL) in the primary (liquid) tank in the presence of liquid injection or continuous condensation. The void fraction should be a positive value between 0.001 and 1.0.

Additional parameters for customizing the twin may be defined on the Additional Input tab. For detailed information on using twins, please see section 3.25 “Modeling Phasic Nonequilibrium using Twinned Tanks” in the SINDA/FLUINT User’s Manual.

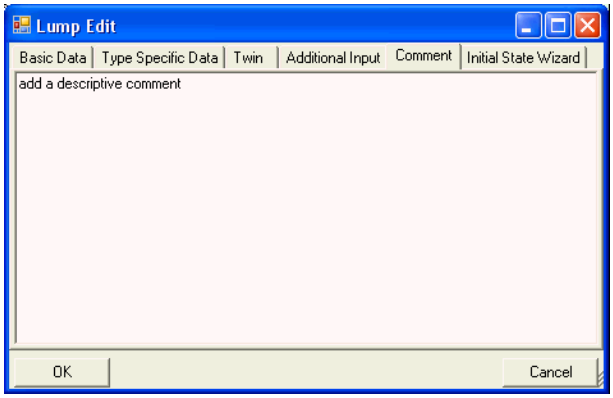
Position:. In the Position tab, the user may optionally define the location of the lump in one, two, or three coordinates. This position is used in combination with the global acceleration vector components ACCELX, ACCELY, and ACCELZ. (These acceleration components may be specified in OPERATIONS logic before a call to a solution is made, or in the Global tab of the Global Control Data edit form, which is accessible on the Advanced tab of the Case Manager.) For example, if ACCELZ has been set to the acceleration of gravity (perhaps using the built-in constants *grav* or *gravsi*),* then specify the Z coordinate of each lump (CZ) as its elevation to model gravitational forces.



Additional Input: As stated earlier, the Lump Edit form and the Object Properties panel only provide access to the most common lump input parameters. The user can define more advanced parameters through the Addition Input tab. Since these inputs are column sensitive and should not start in column 1, Sinaps will automatically start the inputs in column 2. Therefore comments cannot be designated using a ‘C’ in column 1. However in-line ‘$’ comments can be

* Note that in the standard unit set, *grav*=32.174*3600^2 ft/**hr**^{2}, and not the more common 31.174 ft/**s**^{2}. For SI units, *gravsi*=9.81 m/s^2.

used. This form accepts ASCII lump input format as defined in the SINDA/FLUINT User's Manual. If the model uses mixtures or twinned lumps, the additional parameters can be defined by the user in this dialog.

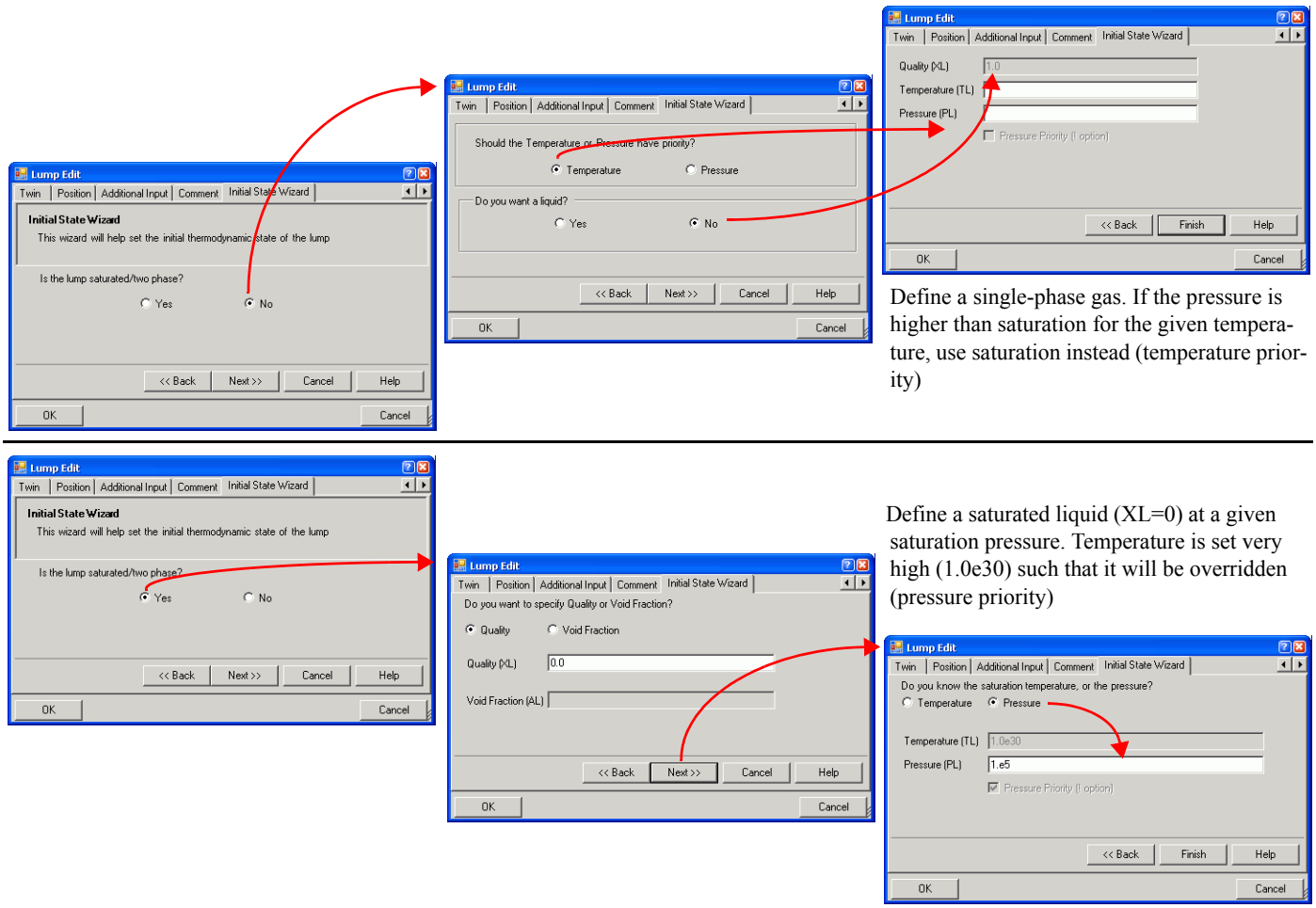


Lump Comment: The comment tab in the lump edit form can be used for documentation purposes.

Lump Initialization Wizard: Sinaps provides a lump initialization wizard available to help the user set up initial lump thermodynamic state. *This wizard is not applicable for use with mixtures.* The wizard is available as the last tab on the Lump Edit form. While intended primarily to be used the first time a lump's state is being defined, its sequence of questions can be repeated as needed to change the initial lump state.

Figure 5-13 depicts two example uses of the wizard. The top half depicts a sequence for a single-phase (liquid or gas), and the bottom half depicts the sequence for a saturated or two-phase state. In the first panel, the user chooses whether the state should be state should be saturated or two-phase, or single-phase (superheated or subcooled, perhaps).

FIGURE 5-13. Lump State Initialization Wizard, Two Examples



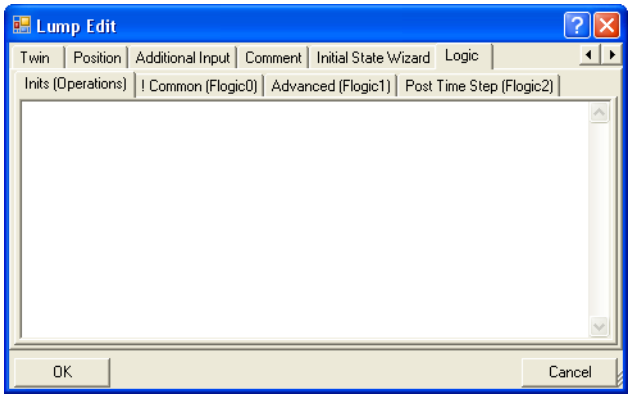
For the single-phase case, the user will be prompted to choose either temperature or pressure to have priority. Since *both* temperature *and* pressure will normally be needed to define a subcooled or superheated state, the purpose of this designation is to clarify what will happen if a value too low or too high (compared to saturation) is entered by the user: which value should be retained? Keep in mind that saturation states will not be known until any FPROP blocks are pre-processed and SINDA/FLUINT is running: Sinaps is not aware in advance of the saturation pressure given temperature (or vice versa), and likely neither is the user.

On this same panel, the user will be prompted to declare whether a gas (superheated vapor) or liquid is desired, which sets the XL designation shown in gray in the third panel: XL=1.0 for gas/vapor, or XL=0.0 for liquid. On the third panel (upper right hand corner of Figure 5-13), the user can finish the form by supplying a temperature and pressure in user units.

For the two-phase case, the user will first be prompted (in the second panel) to choose between quality (gas/vapor mass fraction) and void fraction (gas/vapor volume fraction), though the meanings are the same at the limits of zero (all liquid) and one (all gas/vapor). In the third panel, the user may elect to set either temperature or pressure. (The value of the unspecified parameter will be set to a very high or very low value such that it is sure to be overridden by the saturation condition when SINDA/FLUINT begins execution.)

Note that the user can navigate back to prior forms if they have been entered incorrectly. Upon completion of the wizard, appropriate values of TL, PL, XL, and the pressure priority check box have been set. (If void fraction has been set in the two-phase option, a call to CHGLMP will be generated in the OPERATIONS block of each case.)

Lump Logic. The lump logic tab allows element specific logic to be added. See "Network Element Logic" on page 5-137 for details on how to use element level logic.



5.4.3.3 Multi-lump Editing

To edit multiple lumps, hold down the shift or Ctrl key during the selection process to allow multiple object selection, then right-click on a lump to bring up the object menu. Select Fluid>Edit Lumps. When editing multiple objects, only the input parameter(s) changed during the edit process will be updated. All other input fields will remain unchanged.

5.4.3.4 Changing Fluid Lump Submodel

To move a lump to a fluid submodel, select the lump in the network, right click the mouse to bring up the object menu and select Move to fluid submodel. *All links from the lump to the network will be lost; this option is rarely needed and should only be used as a last resort or with extreme care.* If multiple nodes are selected, all contained links will also be moved to the new submodel and all other links will be lost.

5.4.3.5 Cloning Lumps

Clones are copies of a lump that may be used purely for visual clarity: they do not affect answers. For example, a lump with many paths and ties attached to it can be cloned into two or more copies on the screen, each with its own set of connections. When the model is sent to be solved, only one lump will exist in the model itself. (To verify this, try renumbering a clone: all other copies of that lump will also be renumbered simultaneously.) When post-processed or colored by results, all clones will have the same color.

> Note: When automatic ducts are generated, clones are identified and *excluded from the creation of macros*. When a "Show Created Macros" input check is performed, or the .sin file is written, the macro generator will uncheck the "Aggregate into continuous passage" check box in the lump edit form. If clones are later deleted, this box will remain unchecked until the user reactivates it.

To create a clone, click the right-hand mouse button while hovering over the lump to be cloned, and select clone from the pop-up list.

To delete a clone, delete it as if it were any other lump: the other instances will not be deleted. In other words, deletion applies only to the selected clone, and not to all instances (copies) of that object.

> Tip: A common use of a clone is to make a single lump appear on more than one layer simultaneously. In other words, each clone can be assigned to a distinct layer. Similarly, one clone of a lump might be contained in a collection, while the other is not (or is in a different collection).

5.4.3.6 Setting Lump Defaults

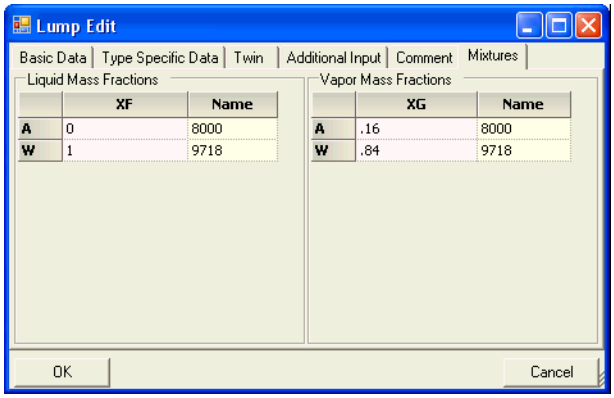
Lump defaults can be set using the Model Browser or the Model Defaults dialog. For more information, see "Setting Network Element Defaults" on page 5-6.

5.4.3.7 Setting Initial Mass Fractions for Mixtures

The initial concentration of each species relative to its phase (liquid or vapor) is required to specify the initial thermodynamic state. These concentrations are specified as mass fractions on the mixture tab of the Lump Edit form (Figure 5-14).

XFx . . . . . Mass fraction of constituent "x" in **liquid** phase, where "x" is A through Z
XGx . . . . . Mass fraction of constituent "x" in **gas** phase, where "x" is A through Z

FIGURE 5-14. Defining Mixture Mass Fractions



Both of these variables represent a fractional partioning of the liquid or vapor phases in each lump. Therefore, the sum of XFx over all constituents "x" will be unity for a liquid-containing lump (and zero otherwise). Similarly, the sum of XGx over all constituents "x" will be unity for a gas- or vapor-containing lump (and zero otherwise). For example, if a lump is defined as XL=0.01, XFA=0.1, XFC=0.9, XGA=0.3, and XGD=0.7, this means that the lump will initially contain 1% gas/vapor by mass, of which 30% is species A and 70% is species D by mass. Of the 99% of the mass that is liquid (1.0-XL=0.99), 10% will be species A (which is apparently a volatile substance) and 90% will be species C by mass.

If a condensible/volatile constituent is specified (e.g., air/water mixtures), the situation can become rather complex because the gas fraction of the volatile species cannot exceed a saturation limit given the temperature and pressure in the lump: there is a maximum XG for a vapor if XF>0 for that species. For air/water (and other psychrometric) mix-

tures, utilities such as HUMR2X are available to help calculate inputs such as XG (absolute humidity) of a condensible phase given relative humidity or dew point. See Section 7.10.2.4 of the SINDA/FLUINT User's Manual for additional information.

5.4.4 Fluid Paths

Paths are network links describing how fluid is moved from one fluid lump to another, usually as a function of the pressure difference between the two lumps. They can only be created between two lumps residing in the same fluid submodel. Paths can represent an assortment of devices such as piping, valves, filters, orifices, pumps, turbines, etc. Paths are subdivided into two types: tubes and connectors.

> ***Tubes*** are paths used to model fluid lines with non-negligible inertia. This means that the flow rate inside a tube cannot change instantaneously—it takes time for the forces acting on the tube (such as pressures at the ends) to accelerate or decelerate the fluid inside the tube. Tubes are governed by a differential equation for momentum that roughly corresponds to Newton's equation of motion: F=ma.
>
> ***Connectors*** are paths with insignificant inertia (flow rates may change instantaneously), and are governed by an algebraic (time-independent) relationships between flow rate, pressure differential, etc. Connectors may be used to simulate pumps,* valves, generic (K-factor style) pressure losses, capillary passages, leaks, orifices, turbines, compressors, constant mass or volumetric flow rates, and short tubes. This last connector type, the STUBE or short tube (also listed as "STube"), is an important element since it is identical to the tube except that it has no inertia. Table 2 provides a summary of the available connector devices.

Paths have a wide variety of type-dependent descriptive parameters associated with them. While most of these parameters are inputs, some (such as mass flow rate or "FR"), are simply initial conditions since the value will be calculated during the run and therefore cannot be changed by the user in logic or expressions once the solution process is underway. The positive direction for FR is defined by the arrow in the network diagram with positive flow moving in the direction the link was created. This only determines the sign convention for a path—flow rates may be zero or negative (e.g., reversed).

TABLE 2. FLUINT Connector Device Models

Name	Toolbar Icon	Description
NULL		No device (user-input GK, HK, EI, EJ, DK). This is very advanced, rarely used option.
STUBE		Small or short (time-independent) tube. Same as a tube, but without inertia (reacts instantaneously).
CAPIL		Capillary (surface tension dominated) passage. For wicks, slots, tubules, grooves, etc.
LOSS		Generic head loss component. For losses in bends, tees, orifices, valves, etc.
LOSS2		Two way generic head loss component. Same as LOSS but direction sensitive.
CHKVLV		Check valve: LOSS element that closes for negative flow rates.

* Despite having an instantaneous pressure-flow response, pumps and other turbomachinery may be assigned a time-dependent shaft speed using a co-solved first-order ordinary differential equation (ODE). Refer to DIFFEQ1 in the SINDA/FLUINT User's Manual for details and examples.

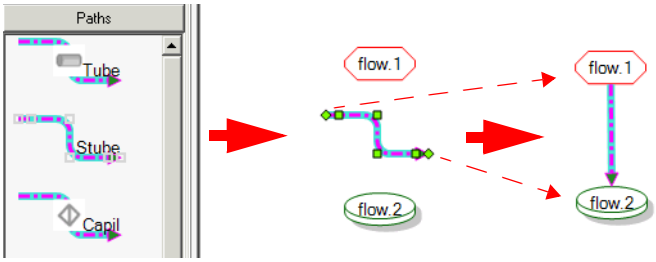
TABLE 2. FLUINT Connector Device Models

Name	Toolbar Icon	Description
CTLVLV		Control valve: LOSS element that user may alter or close according to control logic. If the K-factor is set negative, it closes.
UPRVLV		Upstream pressure regulator valve: LOSS-like element that opens or closes as needed to maintain a desired pressure (PSET) at the upstream lump.
DPRVLV		Downstream regulator valve: similar to the UPRVLV but regulates the downstream pressure.
ORIFICE		Sharp or long orifice, or valve (regulator or opening/closing)
MFRSET		Maintains a constant mass flow rate. For simple pumps, boundary conditions, certain valves (including closed,) etc.
VFRSET		Maintains a constant volumetric flow rate. Similar to MFRSET.
PUMP		Centrifugal pump or fan, partial or full performance curves.
TABULAR		Generalized head/pressure-drop versus flow rate relationship input as tables: performance mapping via interpolation tables
TURBINE		Gas, steam, or hydraulic turbine, partial or full performance curves.
COMPRESS		Axial or centrifugal (variable displacement) compressor.
COMPPD		Piston, vane, scroll (positive displacement) compressor.

5.4.4.1 To Create a Path

To use the toolbar method to create a path, fluid lumps representing the inlet and outlet of the path must be created first; a path's endpoint lumps should exist prior to creating the path. To create a path, click on the desired path type in the fluid toolbar and then place the cursor near the perimeter of the first lump icon. When the cursor changes to a hand ( ), drag a line between the first and second lumps. The direction the path is created defines the positive flow direction: the first lump is the defined upstream end, and the second lump is the defined downstream end. (This direction is a sign convention only, and does not prevent simulation of flow reversal.) Alternatively to draw a path, hover the cursor over the first lump, select and hold the shift key while left clicking on the object. Hold the shift key as the cursor is pulled away from the object and the link is drawn to the second lump.

Alternatively, paths can be created using the Tool Pallet. Either drag the desired path type to the network diagram and then connect the endpoints to the appropriate lumps (as shown below), or click on a path type to select it, and then drag a line between two lumps (i.e., the same as the toolbar method above). Paths will be created based on default parameters. After being created, most paths will require editing since many connector parameters are not defaulted.



Some path types are consolidated under a single icon. For example, the constant mass flow rate (MFRSET) and the constant volumetric flow rate (VFRSET) devices are both referred to as a "SetFlow." Change the path type (mass or volume basis) on the Type Specific Data tab of the Path Edit form.

Up- and downstream pressure regulators (UPRVLV and DPRVLV) are similarly consolidated as "Pressure regulator valves."

5.4.4.2 Editing a Path

A path can be edited using either the Object Properties window or the Path Edit form. The Path Edit form can be accessed by double-clicking on the path. Alternatively, clicking on a conductor will highlight it (grip points will become visible). Once selected, the conductor data is displayed in the Object Properties window and can be edited directly. To edit multiple paths, hold the shift or Ctrl key during multi-object selection and then place the cursor over a path and right-click to access the object menu, then select Edit Path. When editing several paths, only the parameter modified will be updated for each path. The Path Edit form is a tabbed to provide access to various input options as shown in Figure 5-15.

Path Basic Data: The Basic Data tab provides access to define the path identification number, type of path, twin options, duct aggregation (for tubes and Stubes), and other basic parameters such as flow rate, phase suction, flow area, choking options, and duplication factors.

Path ID numbers are automatically assigned when created, although the user can override the ID as long as a unique value is specified.

Twin Path Option - By default, paths are single, homogeneous entities: vapor and liquid flow at the same velocity. Alternatively, tubes and Stubes may be input in permanently matched pairs or *twins*. Twinned paths may be used model slip flow. To twin a path, simply select the Twin Path check box. A twin number will automatically be assigned. By default, paths are *not* twinned.

Twins are nearly identical to each other; they share the same endpoint lumps, same type, etc. However, one of the pair is intended to carry liquid when it is available, and the other is intended to carry gas and/or vapor. These are called the *primary twin* and *secondary twin*, respectively. Twinned paths can be used in conjunction with twinned tanks for mod-

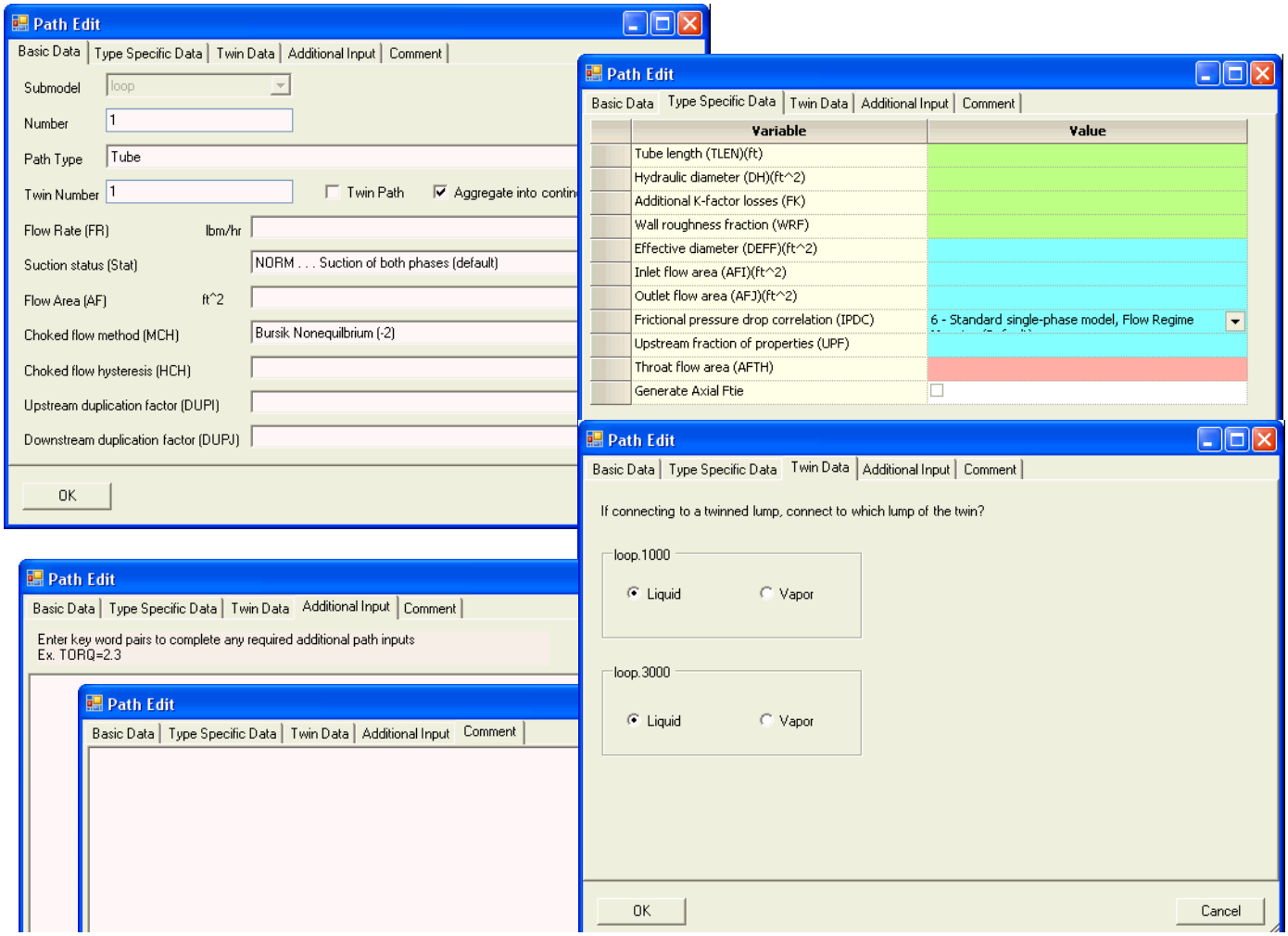
eling nonequilibrium two-phase flow. Twinned paths can also be used without twinned lumps to model slip flow, thus capturing the velocities variations between liquid and vapor in a single path.

> Note: For detailed information on modeling slip flow, please see section 3.17 in the SINDA/FLUINT User's Manual.
>
> For more information on modeling phasic nonequilibrium, please refer to section 3.24 in the SINDA/FLUINT User's Manual.

Aggregate into continuous passage (duct) - This option (for tubes and Stubes only) flags Sinaps to check the path against adjacent lumps and paths to determine if it should be specified as part of a continuous duct macro (see "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5). By default this option is enabled, but it can be disabled by the user if necessary. The automatic macro option can be globally disabled through the preferences dialog, see "Setting Preferences" on page 3-11. In general, *it is recommend that the option be left enabled unless the user needs to locally override the aggregation into an automatic duct*

Flow rate (FR) - This is the initial mass flow rate. It should be chosen carefully (unless FASTIC is called) since it will affect solution speed. Zero flow rates should only be input where they are expected. Small flowrates adversely affect time steps because the flow rate changes per time step are often restricted to a percentage of the total current flow rate. This is an *initial* flowrate for the purposes of initializing the model and it should not be changed during a run.

FIGURE 5-15. Path Edit Form



Suction Status (STAT) - This is the liquid/vapor and constituent suction status flag. The assumption of homogeneous phase distribution in two-phase lumps may be avoided using this option, as can the assumption of homogeneous distribution of species in mixtures. The default value is STAT=NORM, which means that the path will extract the fluid in the upstream lump, without consideration to its phases or species: whatever is in the lump will flow into the path. Under this default condition, a path will extract both liquid and vapor from a two-phase upstream lump, and will extract the homogeneous mixture of constituents from the upstream lump.

The suction status can be set such that the path will extract only liquid or only vapor or gas from such a lump, if that phase is available. This is useful in the simulation of stratified tanks, liquid/vapor separators, and capillary devices. This option provides a pull-down menu of the various choice which are self-descriptive. An important restriction applies to STAT operation: liquid-only or vapor-only suction only applies when the upstream lump is in a two-phase state. Otherwise, the path extracts whatever phase is available even if that phase is not the one specified in the STAT option. This operation may be likened to the draining of a swimming pool: liquid water is extracted through the bottom of the pool, but once all the liquid is gone, the air will flow into the drain. Phase and species suction options state a *preference*, not a *demand*.

Additional STAT options can be used to extract or avoid specific species from upstream tanks, as needed to model non-homogeneous distributions, mass transfer effects such as adsorption, and even chemical reactions. These species-specific options work with the above phase-suction options, and are described separately in Section 3.19.4.

> Note: The STAT option is manipulated internally by twinned paths, and should not be used in combination with them unless they have been forced to stay homogeneous using the NOSLIP routine (see Section 7.10.1.2 of the SINDA/FLUINT User's Manual).

For more information on phase-specific suction, see section 3.11.1 "Phase Suction Options" of the SINDA/FLUINT User's Manual.

Flow area (AF) - This is the cross-sectional flow area for the path (if twinned, it applies to whole passage). If no flow area is given or if AF is negative, AF is automatically calculated for tubes and Stubes based on the hydraulic diameter assuming that the duct is circular. Therefore the flow area need only be input for noncircular cross-sections, at least for tubes and Stubes. For other path types, the flow area should be input.*

Choked Flow Modeling (MCH and HCH) - The speed of sound within a restriction (valve throat, orifice opening, nozzle throat, etc.) or at a low-pressure exhaust cannot be exceeded. Once the fluid achieves that limiting velocity at the restriction, the flow rate and pressure drop characteristics of that path are determined not by its flow resistance, but by the sonic limit: the flow becomes choked. For most loss elements and valves the minimum flow area is much less than the flow area (AF) that is used as the basis for dynamic head and kinetic energy. For sonic limit (chocking) checks, the minimum throat area, AFTH, must be input as a separate parameter.

Choked Flow Method (MCH) - Choking is enabled by default, and it is controlled by the Choked flow method flag. By default, choking is calculated using the Bursik two-phase sound speed method with nonequilibrium expansion (MCH=-2).† This can be changed individually for paths or globally (prior to path creation) through the path defaults, accessible through the Model Browser.

Choked Flow Hysteresis (HCH) - This fraction is *the* amount of hysteresis to be applied between the critical mass flow rate (FRC) and the current mass flow rate (FR). The default hysteresis is 0.02 or 2%, meaning that flow will not begin choking until current flowrate exceeds 102% of the critical flowrate (FR>1.02*FRC), and will not cease choking until it is below 98% of critical flowrate (FR<0.98*FRC). This 4% lag exists purely for numerical stability, even though there is some physical basis for such behavior in nonequilibrium two-phase throat states.

> Note: The default parameters are chosen in order to proceed cautiously: the program checks everywhere for choking, and applies nonequilibrium expansions in doing so. These defaults can frequently stress models and fluid property limits, so knowing when and how to suspend choking calculations becomes an obligation on the user, even in models in which choking is not an issue.

* Even for paths such as SetFlows (MFRSET and VFRSET connectors), specifying flow area is recommended because otherwise the software is unable to track changes in kinetic energy for high-speed flows. Also, the flow areas of adjacent paths should be equal, otherwise an error term will result in the kinetic energy calculation.

† Choking detection and simulation in two-phase flows is a complex topic. Refer to the SINDA/FLUINT manual for details.

Duplication Factors (DUPI and DUPJ) - Paths may be assigned duplication factors, which is a general device for exploiting symmetry to reduce the size of the model. The key to understanding duplication factors is this: *the defined upstream lump "sees" a path DUPI times, while the defined downstream lump "sees" the same path DUPJ times*. For example, an upstream duplication factor (DUPI) of 10.0 means that 10.0 such paths appear from the perspective of the defined upstream lump even though only one path exists. Upstream and downstream duplication factors do not need to be equal. Duplication factors do not need to be integers. Zero values are commonly used to "shut off" a path, which causes the path to be ignored from the viewpoint of the endpoint lumps, even though the path's flow rate will continue to be updated.

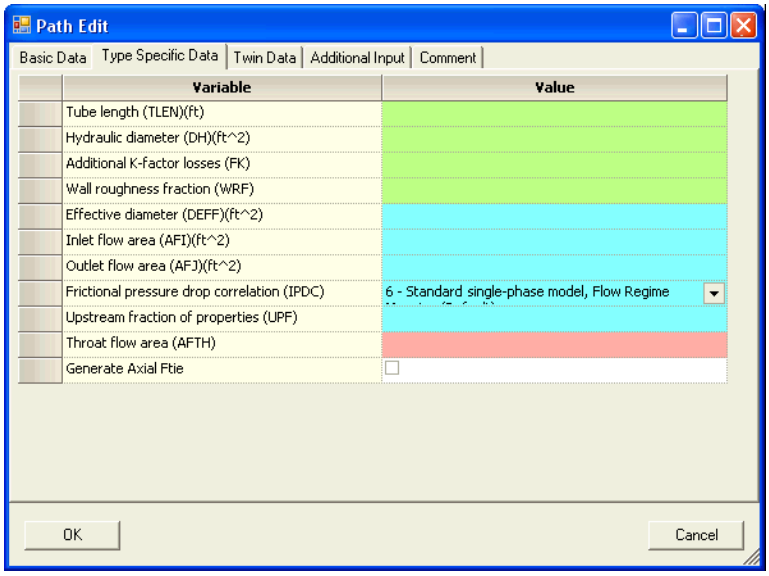
The user must take care to assure that if a duplicated section flows back into the main network, the total upstream and downstream duplication factors balance each other. Otherwise, the virtual subnetworks will act as unintentional sources or sinks of mass and energy, which may result in illegally high or low pressures. Unbalanced duplication factors can results in "ENERGY STABLE BUT NOT BALANCED" error messages, and can also be detected using the SUMD-FLO routine documented in the SINDA/FLUINT User's Manual.

For more information on duplication factors, see section 3.12 "Symmetry and Duplication Options" in the SINDA/FLUINT User's Manual.

Type Specific Data: "Defining Path Type Specific Data" on page 5-44 provides the necessary information required on Type Specific Data tab for each path type. This tab allows the user to define the parameters of the path depending on the type of path selected on the Basic Data tab. The contents of this tab will vary depending on the type of path.

The parameters listed are the most commonly used parameters for each type of device. For detailed descriptions of each path parameter, please refer to section 3.4 of the SINDA/FLUINT User's Manual. Many of the advanced parameters are not listed on the edit form. If a parameter is not listed, it can be input on the Additional Inputs tab. "Additional Inputs" must follow the ASCII text format for path inputs.

FIGURE 5-16. Path Type Specific Form



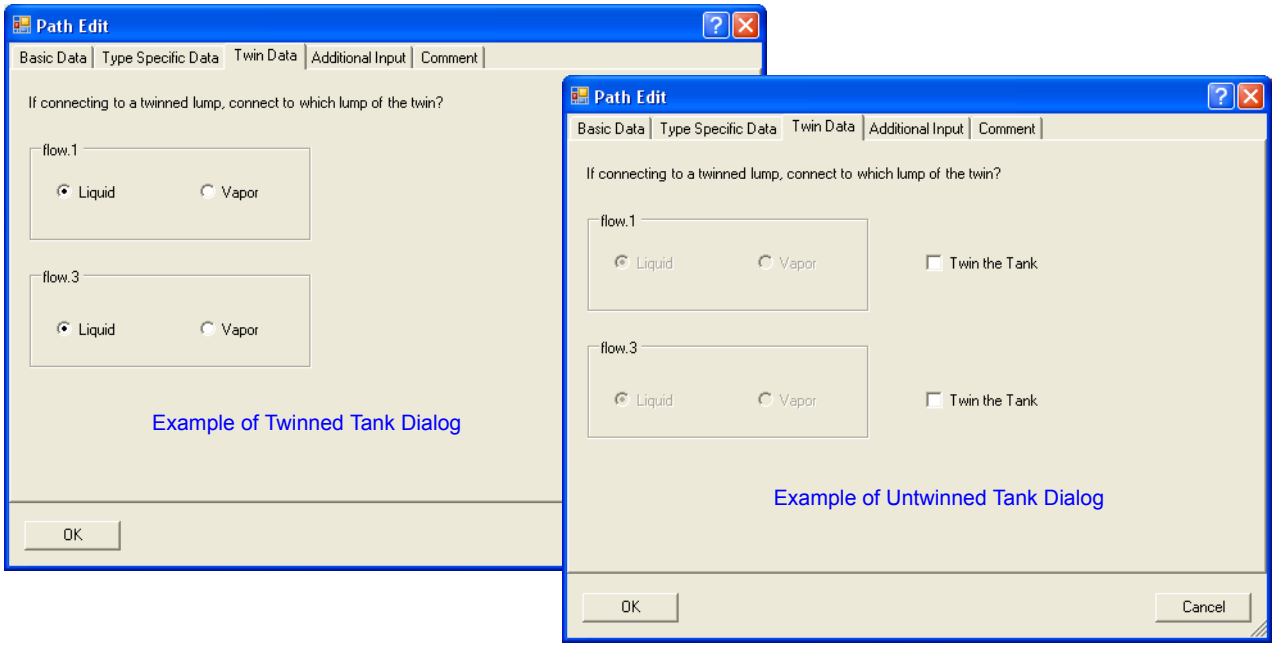
As shown in Figure 5-16, the data fields have been color coded to aide the user in determining what should be set.

Green . . . . . required options or commonly used options
Blue. . . . . intermediate options, not commonly used
Red . . . . . advanced options, rarely used or difficult to use correctly

All path parameters can be input as numeric values, registers, complex expressions, or conditional statements. They may also be based on processor variables (values that change during simulation, such as time, temperature, etc.).

Twinned Data: The Twin Data form shown in Figure 5-17 provides the ability to designate how a path which is *not* twinned connects to twinned lumps. If the path is connected to a junction, a plenum, or is part of a pipe macro, some of all of the form may be grayed out since the dialog only pertains to tanks. When a path is *not* twinned, but is connected to twinned tanks, the user must identify to which side of the twinned tank (liquid vs. vapor twin) each end of the path should connect. If either or both of the adjacent lumps are twinned, the form allows the user to specify the liquid or vapor side of each tank for connecting. If either of the tanks are not twinned the form will provide a checkbox option to twin the adjacent tanks. Checking this box will twin the lump and enable the liquid/vapor selection dialog. (When the twinned tanks are in the homogeneous, single-phase mode, then the path will be moved to the primary tank regardless of this designation, which is applicable only when the tank is in the nonhomogeneous or split mode.)

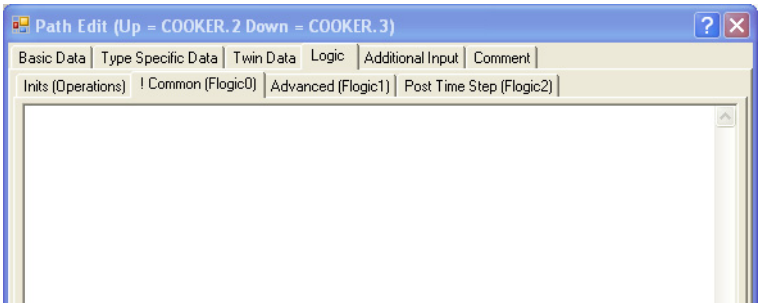
FIGURE 5-17. Specifying Path Connections to a Twinned Lump



Additional Input: The Additional Input tab provides a means for the user to define less commonly used or advanced parameters not available to the user on the Type Specific Data tab. This tab is simply a text input window allow the user to define additional parameters in the standard ASCII text format as defined in the SINDA/FLUINT User's Manual. Since these inputs are column sensitive and should not start in column 1, Sinaps will automatically start the inputs in column 2. Therefore comments cannot be designated using a 'C' in column 1. However in-line '$' comments can be used. Moving a Path (Changing Endpoint Lumps)

Paths can be relocated simply by clicking on the path the highlight and make the grip points visible. Drag the appropriate grip point from the existing lump a new lump in the same submodel.

Path Logic. The path logic tab allows element specific logic to be added. See “Network Element Logic” on page 5-137 for details on how to use element level logic.



5.4.4.3 Setting Path Defaults

Path defaults may be set using the Model Browser or the Model Defaults dialog. For more information, see “Setting Network Element Defaults” on page 5-6.

5.4.5 Defining Path Type Specific Data

This section provides the details required to define each type of path available to the user. For each path, general information and instructions will be provided along with a description of each data field. In most cases, not all parameters are listed in the edit forms. Unlisted parameters are considered advanced or uncommon and can be defined through the additional input tab of the Path Edit form.

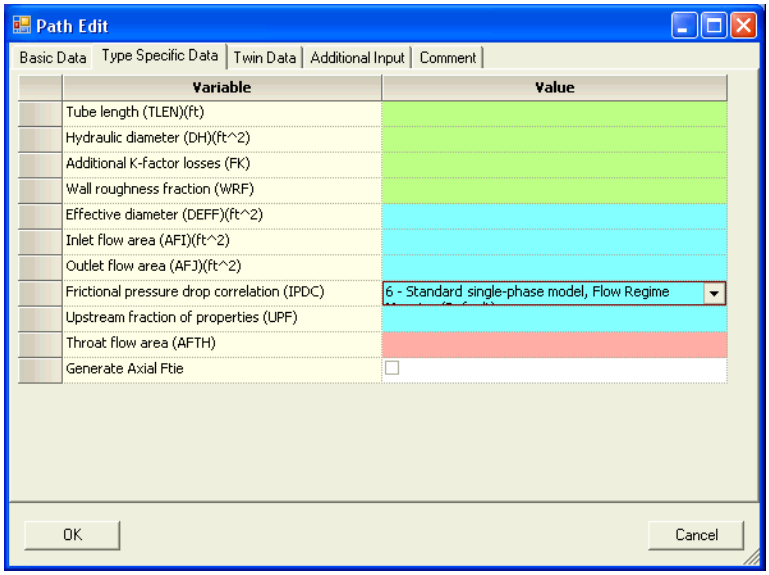
5.4.5.1 Tube and STUBE Input

Tubes are paths used to model fluid lines with nonnegligible inertia. STUBE connectors are paths with negligible inertia.

Figure 5-18 shows the type-specific input for defining a tube or an STUBE. Descriptions of each input are described below. For detailed instructions on the loss devices, please see section 3.5.4 and 3.5.5 of the SINDA/FLUINT User’s Manual.

The data fields highlighted in green indicate required or commonly used parameters, blue indicates less common options and red indicates advanced options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. All input parameters may be defined using registers or expressions.

FIGURE 5-18. Tube and STUBE Type Specific Data Form



Tube Length (TLEN) - The length of the tube. It should be noted that the inertia of the tube is directly proportional to TLEN. This is important information to keep in mind when developing a fluid model, since tube size can directly affect the allowable transient time step. If the TLEN is small compared to the diameter, an STUBE connector should be considered instead of a tube, especially for gas flows. TLEN is also used to determine frictional losses, and (if this tube or STUBE is involved in a heat transfer tie), wetted heat transfer area.

Hydraulic Diameter (DH) - The hydraulic diameter is defined as four times the flow area (AF) divided by the wetted perimeter. This reduces to the diameter for a circular cross section, and to the length of one side for a square cross section. DH is used in calculations such as heat transfer. DEFF (below) defaults to DH, and DEFF is used to determine frictional pressure drops if IPDC is nonzero. If AFI and AFJ are input, and no value of DH is input, then DH and DEFF are initialized as the diameter at the average flow area assuming a circular cross section.

Additional K-Factor losses (FK) - Effects of pressure head losses due to entrances and exits as well as fittings such as bends, tees, expansions, contractions, orifices, valves, etc. may be included as K-factor head losses.

Wall roughness fraction (WRF) - The roughness fraction is defined as the ratio of the characteristic length of the wall roughness to DEFF (usually the same as DH), and is usually on the order of 1.0^{-6} to 1.0^{-3}. If no WRF is given, it is assumed to be zero (smooth wall—this does not mean frictionless). WRF is available in the user logic blocks and may be altered if necessary. WRF is only relevant if IPDC is nonzero, and does not affect friction when the flow is laminar.

Effective diameter (DEFF) - In most cases, the effective diameter will be the same as the hydraulic diameter. The hydraulic diameter assumption is accurate to within about 15% for turbulent flow, but can be off by as much as 40% for laminar flow in extreme cross sections. A corrected or effective hydraulic diameter, DEFF, may therefore also be input. DEFF defaults to DH.

For example, for a square duct, DEFF=0.89DH (where DH is the length of a side). For parallel plates, where DH=2G where G is the gap between the plates, DEFF = 2*DH/3 = 4*G/3. Such factors are based on exact laminar flow solutions.

When DEFF is supplied in addition to DH, DEFF becomes the basis of the Reynolds number for both laminar and turbulent flow frictional calculations. It should also be the basis of the WRF roughness factor supplied independently by the user: WRF = ε/DEFF.

Inlet and outlet flow areas (AFI and AFJ) - If the cross sectional area of the path varies axially, AFI and AFJ can be used to handle that case. Otherwise, AFI and AFJ are left as negative, signalling that AF should be used instead. AFI is the flow area at the defined upstream end of the path, and AFJ is the flow area at the defined downstream end. These definitions do not change if the flow rate happens to reverse during a solution. If both AFI and AFJ are set, then AF becomes the average value. Users are strongly cautioned to add loss factors (e.g., FK) when areas change, especially when the flow area expands in the flow-wise direction.

The flow area should be continuous, or else errors may appear in the kinetic energy terms for high speed flows. “Continuous” means that the flow area of the upstream path (AF or AFJ) should equal the flow area at the downstream path (AF or AFI) for a lump with only two paths. This means that a DH=0.01 tube or Stube should not be upstream of a DH=0.02 path without a reducer or expander between: a tube or Stube whose AFI is not equal to AFJ (in this case, assuming a circular cross section, AFI=0.25*pi*0.01^2 and AFJ=0.25*pi*0.02^2). FLUINT will issue a one-time caution at the start of a run if discontinuities in flow area are detected.

Frictional pressure drop correlation (IPDC) - This flag specifies whether or not FLUINT should automatically perform frictional pressure drop calculations (for FC and FPOW described below), and which correlation should be used for two-phase flow. Setting IPDC=0 means FC and FPOW will not be updated for this tube, and are the responsibility of the user (which is an option that is rarely invoked). Any other value means FC and FPOW will be updated by the standard methods for single-phase flow, and the value of IPDC determines which of several popular two-phase correlations should be used. See Appendix A of the SINDA/FLUINT User’s Manual for further description of these friction models.

The following list details the action taken according to the value of IPDC:

0 . . . . . No FC or FPOW calculations for this tube (user input)
1 . . . . . Standard single-phase model, plus McAdam’s Homogeneous for two-phase
2 . . . . . Standard single-phase model, plus Lockhart-Martinelli’s for two-phase
3 . . . . . Standard single-phase model, plus Baroczy’s for two-phase
4 . . . . . Standard single-phase model, plus Friedel’s for two-phase
5 . . . . . Standard single-phase model, plus Whalley’s recommended choice of 2, 3, or 4 (above) for two-phase
6 . . . . . (Default.) Standard single-phase model, plus flow regime determination and frictional models based on those simplified regimes. Also invokes calculation of FD parameter when used in conjunction with twinned tubes. Flow regime information is also used in two-phase multiple-constituent and nonequilibrium flows to help predict heat and transfer coefficients.

Upstream fraction of properties (UPF) - This parameter is used if IPDC is not zero. This parameter allows the user to calculate FC and FPOW according to modeling preferences. If UPF=1.0, only the properties of the lump currently upstream of this tube will be used for the friction calculations. If UPF=0.0, only the downstream lump properties will be used. If UPF=0.5 (the default), both will be used for an average of the two. Any other UPF (between 0.0 and 1.0 only) will result in a weighted average.

Throat flow area (AFTH) - Flow area for any internal restrictions or contractions that may limit flowrates due to choking (flow cannot exceed sonic velocities in such throats: see MCH and HCH above). If no throat flow area is given or if AFTH is nonpositive, AFTH is assumed equal to AF unless AFI and AFJ are input, and then it is assumed equal to the smaller of those two values. Otherwise, AFTH is independent of AF, AFI, and AFJ. AFTH is available in the user logic blocks and may be altered if necessary. AFTH is only used in the choked flow detection and simulation options.

Generated Axial FTIE - To include heat conduction between the endpoint lumps on the basis of the fluid within the tube or STUBE connector, a parallel axial FTIE can be easily generated by checking this option. The generated FTIE will have the same identifier as the path, and will use the same endpoint lumps. It will also use the same duplication factors. If the endpoint lumps are twinned and so is the path, a pair of twinned fties will automatically be generated using this directive. The use of this ftie option is preferred over independent ftie generation since it will automatically follow rules based on the twin state of the associated lumps and paths.

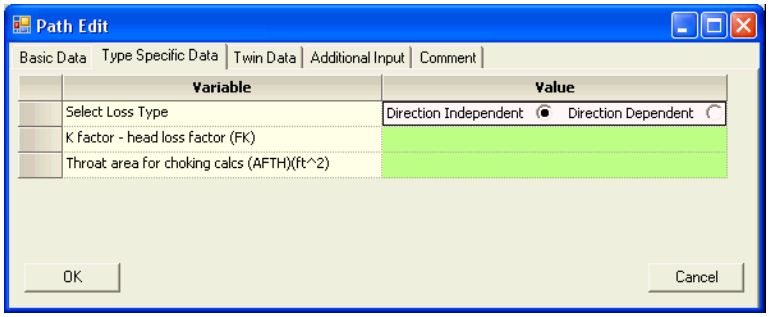
Fties are generally only relevant in very slow moving flows or with highly conductive liquids.

5.4.5.2 LOSS and LOSS2 Input

In addition to frictional losses in pipes, pressure head may be lost due to fittings such as bends, tees, orifices, valves, filters, etc. The loss path option will simulate head losses, which are characterized by a loss factor that is independent of flow direction or allowed to differ according to flow direction.

The loss type-specific form is shown in Figure 5-19. Descriptions of each input are described below. For detailed instructions on the loss devices, please see section 3.5.4 and 3.5.5 of the SINDA/FLUINT User's Manual.

FIGURE 5-19. LOSS and LOSS2 Type Specific Data Form



Loss Type - Defines whether the is the loss is directional independent (LOSS) or dependent (LOSS2). When using the Object Properties task panel to edit a loss, the subtype field controls the type. Subtype0 = direction independent, Subtype1 = direction dependent.

K-Factor head loss (FK) - Effects of pressure head losses due to entrances and exits as well as fittings such as bends, tees, expansions, contractions, orifices, valves, etc. may be included as K-factor head losses. For a directionally dependent loss, FK represents the forward flow (positive) loss factor.

Throat flow area (AFTH) - Flow area for any internal restrictions or contractions that may limit flow rates due to choking (flow cannot exceed sonic velocities in such throats: see MCH and HCH above). If no throat flow area is given or if

AFTH is nonpositive, AFTH is assumed equal to AF. AFTH is available in the user logic blocks and may be altered if necessary. AFTH is only used in the choked flow detection and simulation options.

K factor for negative flow (FKB) - FKB is the K factor corresponding to backward (negative) flowrates. The default for FKB is FK, in which case the connector is indistinguishable from a LOSS device.

5.4.5.3 Check and Control Valve Input (CHKVLV and CTLVLV)

The check valve (CHKVLV) connector has all the same options and operations as does a LOSS connector, except that the valve will close automatically if the flowrate reverses *or if the pressure gradient reverses,* signalling impending flow rate reversal. The positive flow rate direction for the connector is assumed to be the nominal (open) flow direction.

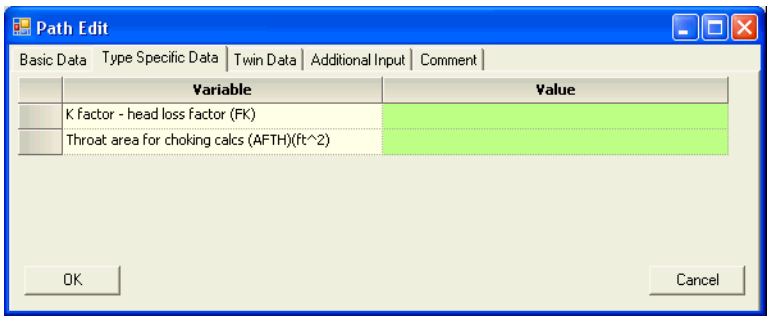
A control valve (CTLVLV) has all the same options and operations as does a LOSS connector except that the user may close the valve by defining a negative FK. The valve may be opened and closed as desired by varying the FK factor, but FK can never be zero (i.e., there must exist some head loss across the valve, however small, when it is not shut). In many engineering applications, the performance of a control valve is given in terms of "Cv." This parameter is specific to English units and is usually used for water. To convert to unitless FK, use the following relationship (where AF is the valve area *in standard FLUINT English units* of ft^2):

$$FK = 3.0E7 * (AF/CV)**2$$

Both the check valve and the control valve will behave like a LOSS element when open (i.e., FK is positive), and like an MFRSET connector with zero flow rate when shut. For this reason, a shut valve has all the same restrictions as an MFRSET or VFRSET connector (see SetFlow below).

The type specific edit form for the Check and Control Valve device is shown in Figure 5-20. Descriptions of each input are described below. For detailed instructions on the check and control devices, please see section 3.5.6 and 3.5.7 of the SINDA/FLUINT User's Manual.

FIGURE 5-20. Check and Control Valve Type Specific Data Form



K-Factor head loss (FK) - Amount of head loss in the valve when fully open. The FK factor will only be applied to forward (positive) flow for a check valve.

Throat flow area (AFTH) - Flow area for any internal restrictions or contractions that may limit flowrates due to choking (flow cannot exceed sonic velocities in such throats). If no throat flow area is given or if AFTH is nonpositive,

AFTH is assumed equal to AF. AFTH is available in the user logic blocks and may be altered if necessary. AFTH is only used in the choked flow detection and simulation options.

5.4.5.4 Pressure Regulator Valves (UPRVLV and DPRVLV)

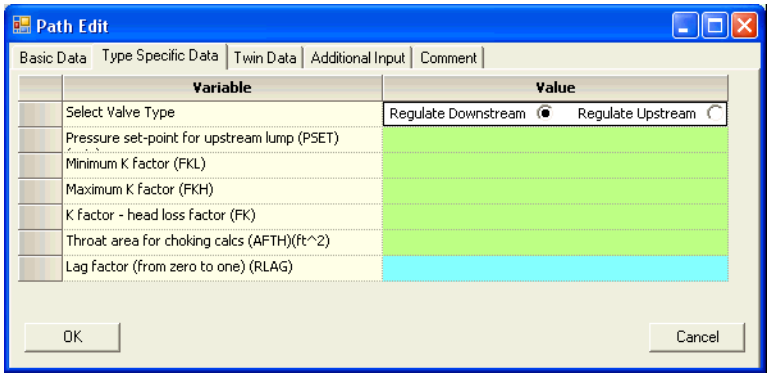
Pressure regulator valves are similar to the LOSS element, except that the K-factor resistance (FK) is not set by the user, but rather is calculated by the program as needed to maintain the pressure at the defined upstream (UPRVLV) or downstream (DPRVLV) lump to be a set point pressure.* Pressure regulators can also be modeled using a controlled orifice (the preferred method), for more details see "Orifice Device (ORIFICE)" on page 5-50. The controlled orifice option provides lower and upper limits on the value flow area along with a time constant which makes the device more reliable and less prone to occilations and instabilities compared to the UPRVLV or DPRVLV devices.

The user may place upper and lower limits on the K-factor, and may set its *initial* value. Otherwise, the FK value becomes an output variable available for inspection but which cannot be directly specified by the user.

The type specific edit form for the pressure regulator device is shown in Figure 5-21. Descriptions of each input are described below. For detailed instructions on the pressure regulating devices, please see section 3.5.8 and 3.5. 9 of the SINDA/FLUINT User's Manual.

The data fields highlighted in green indicate required or commonly used parameters while blue indicates less common options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. All input parameters may be defined using registers or expressions.

FIGURE 5-21. Pressure Regulator Valve Type Specific Data Form



Valve Type - Select if the valve is to regulate the upstream (UPRVLV) or the downstream (DPFVLV) pressure. When using the Object Properties task panel to edit a pressure regulating device, the subtype field controls the type. Subtype0 = downstream pressure regulation, Subtype1 = upstream pressure regulation.

Pressures set-point (PSET) - The pressure to be maintain at the upstream (UPRVLV) or the downstream (DPFVLV) lump, in user pressure units.

* This control is not perfect: it is not implicit. Rather, it is intended to help the user avoid writing their own K-factor adjusting logic. PRVLV options are useful for a system-level representation of a regulator, and more detailed methods are available for specific and detailed regulator modeling.

Minimum K factor (FKL) - Optional minimum K-factor allowed through connector (resistance when fully opened).

Maximum K factor (FKH) - Optional maximum K-factor allowed through connector (resistance when fully closed). Caution: K-factors become undefined for small or zero flows, so consider other methods besides a huge K-factor when modeling a closed valve, including setting DUPI=DUPJ=0.0 in logic or via input expressions.

K-Factor losses (FK) - This is the *initial* K-factor representing the pressure head loss within the valve. The code will seek the correct resistance iteratively.

Throat flow area (AFTH) - Flow area for any internal restrictions or contractions that may limit flowrates due to choking (flow cannot exceed sonic velocities in such throats: see MCH and HCH above). If no throat flow area is given or if AFTH is nonpositive, AFTH is assumed equal to AF. AFTH is available in the user logic blocks and may be altered if necessary. AFTH is only used in the choked flow detection and simulation options.

Lag factor (RLAG) - Lag factor (from zero to one, inclusive). The maintenance of the desired pressure is not perfect nor immediate. By default (RLAG=0.0), it will seek and maintain the required resistance as fast as possible, but inevitably there will be some delay, lag, and overshoot (or undershoot). If needed for stability, additional lag can be added by using nonzero values of RLAG: the higher the value, the greater the lag. A value of 1.0, for example, means infinite lag: no changes to the valve position. Generally, lag should only be added as needed to assist convergence of steady state runs, or to avoid jittery responses in transients if junctions or hard-filled tanks are used at the end-points.

5.4.5.5 Orifice Device (ORIFICE)

The orifice connector can be used to model a square or circular orifice flow restriction or a partially open valve (such as a regulator or a valve opening/closing). Orifices can also be used to model a partially closed valve (needle valve, pressure or backpressure regulating valve, even a fast-acting solenoid valve in mid stroke).

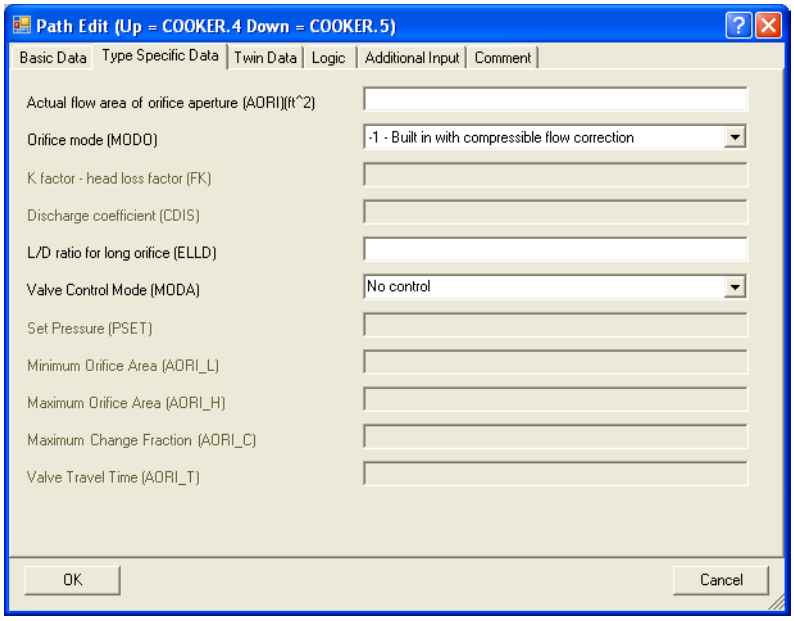
The user must supply both the upstream flow area AF and the opening (hole) area AORI. Although the upstream and downstream flow areas are assumed equal, and although compressibility effects may be optionally included, the calculated K-factor (FK) uses the upstream density for the dynamic head basis. Compressibility effects may be optionally included in the calculation. The default calculation is to use a built-in correlation. As an alternative, the user may specify the discharge coefficient or the K-factor. In all three cases, the size of the vena contracta is estimated and applied as the throat area (AFTH) for choking calculations.

In the control mode (MODA non-zero) the orifice hole size AORI is user-defined, and can be updated during the run. Setting MODA to positive values will provide first-order simulations of an opening or closing valve, while setting it to a negative value will simulate a backpressure (upstream pressure) regulator, or a (downstream) pressure regulator.

For detailed instructions on the orifice device, please see section 3.5.10 of the SINDA/FLUINT User's Manual

The type specific edit form for the orifice device is shown in Figure 5-22. Descriptions of each input are described below. Input fields will activate/deactivate based on which MODO and MODA options selected. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. All input parameters may be defined using registers or expressions.

FIGURE 5-22. Orifice Type Specific Data Form



Actual flow area of orifice aperture (AORI) - Minimum opening area of the orifice. This should be less than the upstream flow area (AF).

Orifice mode (MODO) - Integer mode flag to identifying how the vena contracta will be determined.

MODO=1 . . . . Use built-in sharp or long orifice correlation
MODO=2 . . . . User input CDIS
MODO=3 . . . . User input FK
A negative MODO signals the use of a compressible flow correction.

K-Factor losses (FK) - User defined loss factor. Use this input only if |MODO|=3.

Discharge coefficient (CDIS) - Discharge coefficient. Use this input only if |MODO|=2.

L/D ratio for long orifice (ELLD) - Ratio of hole length to diameter for a long orifice. This is an optional input if |MODO|=1.

Valve Control Mode (MODA) - AORI control mode (integer). Note that a sixth option appears in the pull down list, Enter an expression here. Since MODA can be defined as a register or an expression, this option can be selected to enter an expression the MODA field.

MODA=0 . . . . no control (default)
MODA=1 . . . . opening, to maximum (AORI_H)
MODA=2 . . . . closing, to minimum (AORI_L)
MODA=-1. . . . controlling defined upstream pressure (PL) to PSET
MODA=-2. . . . controlling defined downstream pressure (PL) to PSET

Set Pressure (PSET) - Pressure set point used for upstream or downstream control defined in user's pressure units (static or total according to lump controlled). In logic, PSET is single precision. PSET is only applicable for negative MODA (pressure regulator mode).

Minimum Orifice Area (AORI_L) - Lowest value of AORI allowed. Cannot be less than 1.0e-15 unless zero. Must be less than AORI_H.

Maximum Orifice Area (AORI_H) - Highest value of AORI allowed. Cannot be greater than AF, nor less than AORI_L.

Maximum Change Fraction (AORI_C) - Maximum change in AORI from AORI_L to AORI_H that can occur in one time step or one steady state iteration. May be 1.0 (100%) for no added change lag. AORI_C is active in both steady state and transient simulations

Valve Travel Time (AORI_T) - Time constant for fastest motion from AORI_L to AORI_H or reverse. May be zero for no added time lag. Negative signals fastest rate be used. AORI_T is only active in transient simulations.

5.4.5.6 Capillary Device (CAPIL)

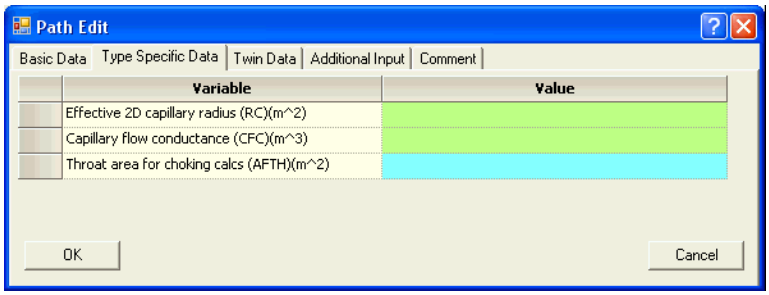
The CAPIL device is used where surface tension is a dominant force in a small passageway. Liquid may always pass through the CAPIL, but vapor and gas cannot pass if the capillary limit (or bubble point) of the passage is not exceeded and any liquid is present to block the passage of vapor. All flow through the connector is assumed to be laminar because of the typically small effective diameter of such devices. There is no preferred direction, blockage may occur in either direction depending on conditions. CAPIL connectors may be used to model wicks, grooves, tubules, thin slots, etc. They might be used as part of a model for gas traps, liquid acquisition devices, liquid/vapor separators, etc.

There is one restriction on the use of this device: the STAT flag is disabled. This is a result of the internal use of liquid/vapor suction action to simulate the device. Also, it should be noted that a multi-element model exists (see “Capillary Evaporator Pumps (CAPPMP)” on page 7-1) which performs the same basic function as CAPIL, but allows heat addition/rejection and therefore can simulate capillary pumping. CAPIL connectors can stop vapor flow but cannot simulate pumping.

Figure 5-23 shows the CAPIL device type specific edit form. Descriptions of each input are described below. For detailed instructions on the CAPIL device, please see section 3.5.3 of the SINDA/FLUINT User’s Manual.

The data fields highlighted in green indicate required or commonly used parameters while blue indicates less common options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. All input parameters may be defined using registers or expressions.

FIGURE 5-23. CAPIL Type Specific Data Form



Effective 2D capillary radius (RC) - RC is the effective capillary radius of the meniscus if it were two-dimensional. For circular tubules, RC is simply one half of the diameter. For a uniform (isotropic) wick, the capillary radius should be available from test data, vendor data, or estimates. For a thin slot, RC is equal to the slot width. Nonpositive RC values are interpreted as perfect capillary devices capable of withstanding or producing an infinite pressure gradient. This is useful for preliminary design work or for controlling steady-state solutions to prevent deprime.

Capillary flow conductance (CFC) - CFC is the capillary flow conductance term, used to find the effective flow resistance of the path depending on the fluid and the phase of the fluid within the path. For circular tubules, RC is simply one half of the diameter. CFC is then calculated using:

CFC = n * AF * DH**2 / (32.0 * TLEN)
where:
n . . . . . Number of tubules in parallel (duplication factors may also be used for multiple tubules, as described in a later section)
AF . . . . . Flow area of each tubule
DH . . . . . Hydraulic diameter of each tubule
TLEN . . . . . Length of each tubule

For a uniform (isotropic) wick, the capillary radius should be available from test data, vendor data, or estimates. The same is true for the properties defining CFC. These are:

CFC = A * P / X (flat plate)
CFC = 2.0 * π * P * Y / ALOG(R_o/R_i) (cylinder)
where:
A . . . . . Wick total frontal area (perpendicular to the flow direction)
P . . . . . Wick permeability (units of area)
X . . . . . Wick thickness (in direction of flow)
Y . . . . . Wick length (perpendicular to the flow direction)
R_o . . . . . Wick outer radius
R_i . . . . . Wick inner radius
ALOG . . . . . Natural logarithm

For a thin slot, RC is equal to the slot width W (because of the one-dimensional meniscus). CFC is then:

CFC = A * W**3 / (12.0 * X)
where:
A . . . . . Slot breadth (perpendicular to the flow direction, equals frontal area divided by W)
W . . . . . Slot width (gap) size
X . . . . . Slot length (in flow direction)

Throat flow area (AFTH) - Flow area for any internal restrictions or contractions that may limit flowrates due to choking (flow cannot exceed sonic velocities in such throats). If no throat flow area is given or if AFTH is nonpositive, AFTH is assumed equal to AF. AFTH is only used in the choked flow detection and simulation options.

5.4.5.7 SetFlow Devices (MRFSET and VFRSET)

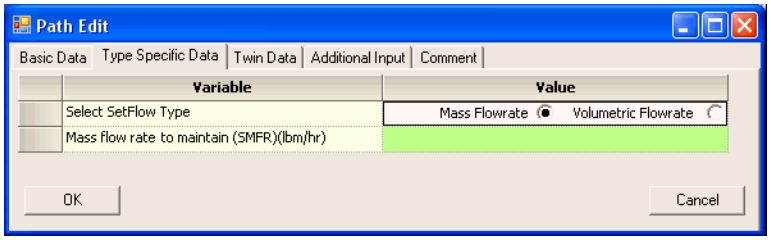
The SetFlow connector maintains the given flow rate through the connector, independent of pressure gradient or fluid state. The SetFlow device can be used to control the mass flowrate (MFRSET) though the path or the volumetric flowrate (VRFSET). For pressure versus flow type devices, a Tabular object should be used instead.

The SetFlow device is a powerful modeling tool. It can be used to simulate an ideal pump or flow control device. It can even be used to control boundary conditions for open systems, or it can represent a completely closed valve. However, it must be used carefully or fatal errors will result. For example, using two SetFlow connectors in the same series line with different flow rates will eventually (if not immediately) cause an error. The user should be careful to avoid similar overspecifications in more complicated networks with parallel passages: at least one flow passage should be allowed to adjust itself without a SetFlow prescribing the flow rate. For this reason, SetFlow connectors cannot be the only type of path attached to a junction.

VFRSET it is a better approximation of a positive displacement pump, and has different behavior than MFRSET when used in compressible flows. Note that the mass flow rate through a VFRSET connector will drop considerably if there is any vapor in the inlet because of the density change.

Figure 5-24 shows the SetFlow device type specific edit form. Descriptions of each input are described below. For detailed instructions on the mass and volumetric flow rate devices, please see section 3.5.11 and 3.5.12 of the SINDA/FLUINT User's Manual.

FIGURE 5-24. SetFlow Type Specific Data Form



Select SetFlow Type (MFRSET or VFRSET) - Select the type of SetFlow device to be modeled. When using the object properties task panel to edit a SetFlow device, the subtype field controls the type. Subtype0 = mass flowrate, Subtype1 = volumetric flowrate.

Flowrate to be maintained (SMFR/SVFR) - The user must specify either the mass (SMFR) or the volumetric (SVFR) flow rate to be maintained. This flow rate is accessible in user logic blocks for inspection or modification.

5.4.5.8 Tabular Device (TABULAR)

This device allows users to specify flow rate versus head (or pressure drop) relationships in tabular formats. The flow rate can be either volumetric or mass flow rates, perhaps corrected or based on a reference state. The head can be either a true head (in units of length, and therefore dependent on the current fluid density) or can have units of pressure drop or pressure ratio.

In a PUMP connector (Section 5.4.5.9), the positive head (H) is understood to be a pumping force in the flow-wise (positive flow rate) direction, whereas in the TABULAR connector the head H is understood to be a retarding force in the positive flow direction: a head loss. In both cases, negative values of H are unusual but by no means illegal. Unlike a PUMP connector, for a TABULAR connector more generalized meanings of G and H are available that are better suited for compressible flows. For example, H might be absolute pressure ratio in stead of difference, and G might be corrected or equivalent mass flow (containing corrections for inlet temperature and pressure).

For each TABULAR connector device, the user may choose one of four ways to define the structure and format of the H vs. G relationship:

4. HG: A doublet list of H values as a function of G values: G1, H1, G2, H2, ...
5. GLIST and HLIST: Paired singlet lists of G (G1, G2, ...) and H (H1, H2, ...) values.
6. HGTABLE: A bivariate table of H as a function of G and some other user variable O. In other words, for each value of O a head-flow rate curve, H(G), exists. O is completely user defined. For example, it might be the open fraction for a valve, or the inlet temperature for a viscosity-dependent device or control valve. The current value of O is “OFAC,” and may be specified in logic blocks or expressions. For example, “flow.ofac102” is the O factor for TABULAR connector #102 in fluid submodel *flow*.
7. GHTABLE: A bivariate table of G as a function of H and some other user variable O. In other words, for each value of O a head-flow rate curve, G(H), exists. This option is an inversion of the HGTABLE option.

In all cases, adjacent H values cannot be equal. By default, the G value must monotonically increase, but H need not be monotonic. If instead H is monotonic, this can be specified using the MONOH=YES option described below, which then permits (but does not require) G to be nonmonotonic. (“MONOH=YES,” if used, must be added to the Additional Input tab.)

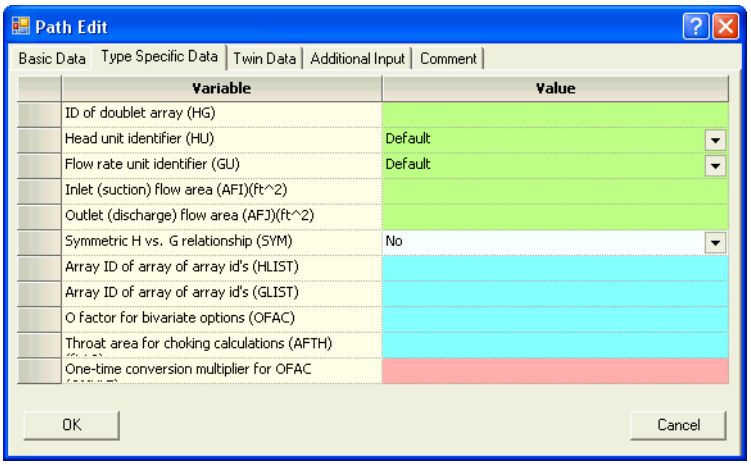
By default, the H vs. G curves are assumed to be symmetric (SYM=YES). This means that a flow reversal will also cause a head loss in the opposite direction. Therefore, using the symmetric designation prevents the user from having to supply curves in the negative flow rate region since the device is assumed to operate equivalently in both directions. If this is not the case, then set SYM=NO. In this case, the user should supply H and G values for negative Gs (and usually with negative Hs). Otherwise, FLUINT will extrapolate from available points into the negative flow rate region. Indeed, *operation outside of the supplied data values will always be treated using extrapolations, whatever values have been supplied and whatever suboptions have been used.*

The type-specific edit form for the tabular device is shown in Figure 5-25. Descriptions of each input are described below. For detailed instructions on the tabular device, how to add torque and shaft power or dissipation, along with other advanced options including use of inlet reference states and modeling of gas labyrinth seals, please see section 3.5.14 of the SINDA/FLUINT User’s Manual.

The data fields highlighted in green indicate required or commonly used parameters, blue indicates less common options and red indicates advanced options. Unless stated otherwise, all parameters are available in the user logic

blocks and may be altered if necessary. Most parameters may be defined using registers or expressions, excepting array IDs.

FIGURE 5-25. Tabular Type Specific Data Form



ID of doublet array, H vs. G (HG) - Input either the Table ID or SINDA/FLUINT array number containing doublet array G1, H1, G2, H2, ... G must increase by default, and adjacent H values cannot be equal. If SYM=YES (default), then G1=H1=0.0 (unless HU=201, in which case H1=1.0). If MONOH=YES is added to Additional Input, then H must increase monotonically, and G can optionally be nonmonotonic. Alternatively, the use may input two singlet arrays and reference them with HLIST and GLIST (for more information see section 3.5.14 of the SINDA/FLUINT User's Manual.)

Head unit identifier (HU) - Integer flag for head units: see Table 3. In addition to those options, HU=201 may be used to designate H as referring to pressure *ratio* (defined inlet to outlet) instead of pressure *drop*. HU=202 may be used to designate H as referring to the Martin β factor, which is a function of pressure ratio and ENLT (specified separately in Additional Input). The meaning of H can be further modified via the ISTP option.

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 4. The meaning of G can be further modified via the ISTP, NAF, and MREF options (supplied in Additional Input if used).

Inlet flow area (AFI) - Inlet flow area. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically.

Outlet flow area (AFJ) - Outlet flow area.

Symmetric H vs. G relationship (SYM) - Symmetric H vs. G relationship flag. If SYM=YES (the default, except when HU=201, and mandatory when HU=202), then the first H and G values should be zero (the first H must be unity for HU=201), and these values will be applied to both flow rate directions. Otherwise, if SYM=NO, then the G values will normally start with negative values (with the initial H values usually negative as well) to cover this region.

Array ID of array of array id's (HLIST) - Input either the Table ID or SINDA/FLUINT array number containing singlet array H_1, H_2, ... with each value corresponding to the GLIST array. Adjacent H values cannot be equal. If SYM=YES (default), then H_1=0.0

Array ID of array of array id's (GLIST) - Input either the Table ID or SINDA/FLUINT array number containing singlet array G_1, G_2, ... G must increase monotonically by default. If SYM=YES (default), then G_1=0.0

Throat area for choking (AFTH) - Throat flow area for choking detection and calculations. If no throat flow area is given, AFTH is assumed equal to AF. Often, choking behavior is built into the H versus G tables, in which case the user should set MCH=0, which renders the value of AFTH irrelevant.

O factor for bivariate options (OFAC) - This parameter is for bivariate options (HGTABLE or GHTABLE). It is a user defined value that can be used to interpolate into families of head curves that are input under the Additional Input. The value of OFAC is then used to interpolate into these tables to produce a head versus flow curve at any value of OFAC. Negative OFAC signals closure of the connector whether or not the HGTABLE or GHTABLE options are used. Any nonnegative value will reopen the path.

One-time conversion multiplier for OFAC (OMULT) - One-time conversion multiplier for O units in array: every O value in the bivariate arrays (HGTABLE or GHTABLE) will be multiplied by OMULT to match the model units of OFAC. (If multiple TABULAR devices use the same HGTABLE or GHTABLE array, they all may use distinct values of OMULT without conflict.)

5.4.5.9 Pump/Fan Device (PUMP)

The PUMP connector is a model of a fan or a centrifugal, axial, or mixed-flow pump: a pump whose performance varies as a function of volumetric flow rate for a given rotational speed. In order to simulate such a pump, the user must specify pressure head as a function of volumetric flow rate and perhaps rotational speed. The PUMP device can be used to specify a simple fan/pump curve or a full performance map at many speeds. To specify a fixed flow, the SetFlow device ("SetFlow Devices (MRFSET and VFRSET)" on page 5-54) should be used instead of the pump device.

Throughout the PUMP options, users may elect to use dimensional head and flow rates to define performance parameters, or (by using COEFS=YES) they may instead elect to use nondimensional or quasi-nondimensional (depending on the values of HCF and GCF) head and flow rate *coefficients* ($\Psi$and Φ). The choice of whether or not to use coefficient-based performance parameters can vary from pump to pump (if more than one exist in the model), but otherwise the rule is all-or-nothing within a single pump: either use H and G, or else use Ψ and Φ, but the two forms cannot be mixed within a single device.

In an ideal pump, "similarity rules" apply: the H scales perfectly with SPD^2 and the G scales linearly with SPD (with other variations fixed, such as pump rotor diameter). If these rules hold true, then a single curve of Ψ-Φ completely defines the map.

There are various ways to represent the head-flow performance map depending on information available:

1. If the pump similarity rules apply, the user can either of the options below. These methods will be collectively referred to as "the HG= method."
    a. Specify a single Ψ-Φ curve (with COEFS=YES, and HCF and GCF specified along with the units of the underlying H and G values

   b. Specify a single H-G curve along with a reference speed (RSPD) upon which that curve is based (along with the default COEFS=NO and the units of H and G specified). For pumps with no speed (SPD) variation, RSPD is optional.

2. Otherwise, if the similarity rules either don't apply or don't apply perfectly or globally, then the user can elect to input a full map: a series of Ψ-Φ or H-G curves at various speeds. This method will be referred to as "the HLIST= method." (In the case of Ψ-Φ maps, an option exists to use a single curve of Φ with only Ψ being dependent on SPD.) When supplying a full map, the points in the curves are assumed to be corresponding at each speed, meaning that flow coefficients (whether or not coefficients are used in the map itself) are at least roughly equal. This requirement, which will be described in more detail later, is met by the output generated from most pump analysis codes. This method will be referred to as "the HLIST= method."
3. If the Ψ-Φ or H-G maps exist as functions or are provided via calls to (perhaps COM-linked external) pump codes, then the current value of H and G (or Ψ and Φ) can provided directly. This method will be referred to as "the GPMP= method."Although data is often perceived as H(G) ... head as a function of current flow ... SINDA/FLUINT actually requires the reverse: G(H) or Φ(Ψ): flow as a function of current head. Furthermore, SINDA/FLUINT requires the slope at the current SPD and head: dG/ dH or dΦ/dΨ, which cannot be zero. In other words, the user can omit pump arrays and can instead manipulate the pump parameters directly in logic and/or expressions. The variables for H and G (or Ψ and Φ if COEFS=YES) are HPMP and GPMP. The code provides the current value of HPMP (head or head coefficient), and the user must supply the corresponding values for GPMP (flow rate or flow coefficient) and DGDH (the slope). *When using the GPMP= method, automatic corrections such as two-phase, cavitation, and viscosity are omitted: they are assumed to have already been taken into account by the user in the provided values of GPMP and DGDH and the efficiency EFFP.*

Pump efficiencies are optional but should be specified or at least estimated. There are three methods for defining efficiency. For detailed information on defining efficiencies, please see section 3.5.13.2 of the SINDA/FLUINT User's Manual.

1. The user can define a specific value using EFFP ("the EFFP= method").
2. The user can define efficiency as an interpolated array of efficiency vs. G (or Φ if COEFS=YES): "the AEFFP= method."
3. If a full map of H-G or Ψ-Φ has been provided via the HLIST= method, a full map of efficiencies can be provided in parallel. This method is known as "the ELIST= method."

The code calculates the hydraulic torque as:

```
TORQ    =  TCF*|GPMP|*HPMP/(EFFP*SPD)
```

where GPMP and HPMP are in units defined by COEFS, GU, and HU, and TCF is input by the user to perform whatever unit conversions are necessary. TORQ is a translatable path output variable. TCF defaults to unity.

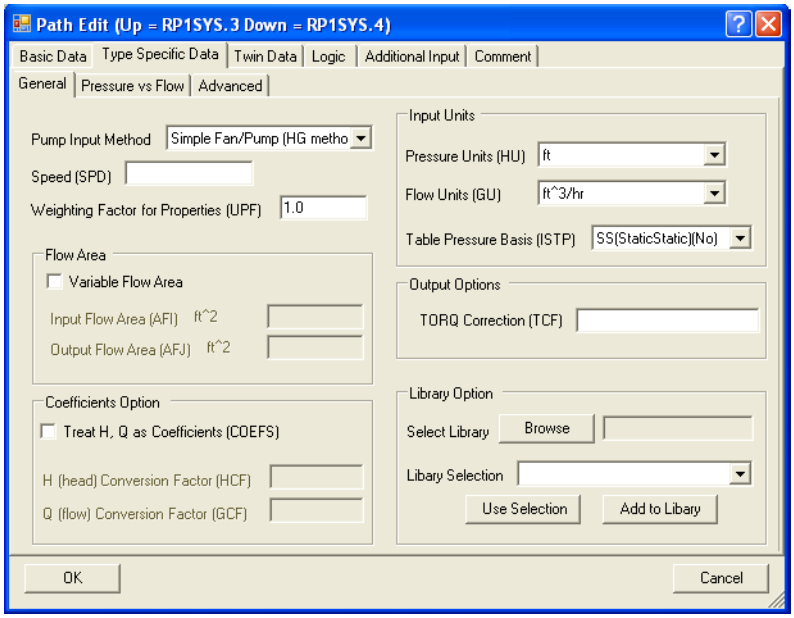
Hydraulic torque may be used along with co-solved ordinary differential equations (e.g., DIFFEQ1) to calculate the current speed of the pump during transients, or may be used to solve a torque balance equation for steady-states (e.g., for turbopumps). Note that TORQ is positive for energy into the fluid system, so the negative of TORQ should be set proportional to dω/dt, the rate of change of shaft speed.

Figure 5-26 shows the type-specific edit form for a pump. Inputs are organized on three tabs, General, Pressure vx Flow, and Advanced. Descriptions of each input are described below. For detailed information on describing pumps, cavitation detection and simulation, torque, heating, and other advanced options, please refer to section 3.5.13 of the SINDA/FLUINT User's Manual.

Note: The Basic Data tab of the Pump form has an option to "Use 5.3 Type Specific Data". The Pump form has been redesigned in version 5.4 to provide increased "user friendliness", assisting the user in determining which inputs are required for a pump and providing access to Sinaps Tables for array data input. When a 5.3 or older model is opened, the "Use 5.3 Type Specific Data" option will be checked and the Type Specific Data tab content will appear has the heritage style input form. The user may continue using the heritage form however it is recommended to update to the new pump form by unchecking the box. Be aware that when updating to the new form, some data (such as performance curves) will need to be reinput. For information on the heritage style form please see "Heritage

Pump General Tab. Data input on the General Tab applies to all pumps regardless of the pump input method. For additional information on pumps, see"Pump/Fan Device (PUMP)" on page 5-57.

FIGURE 5-26. Pump General Form



Pump Input Method - A pump can be defined as either a "Simple Fan/Pump" using a single curve to defined pressure vs flow (HG method), or a "Fully Mapped Pump" using a family of curves to define the pressure vs flow at various speeds. The input method selection controls the content of the Pressure vs Flow tab so only applicable data fields will be displayed.

Speed (SPD) - This is the current/initial pump speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency.

Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density. Note that ANSS etc. are always based on current inlet (suction) properties.

Variable Flow Area - By default the flow area is defined as a constant area as defined by the field Flow Area (AF) on the General tab of the pump form. Checking the Variable Flow Area box enables a variable flow area across the path. When checked both the inlet (AFI) and outlet (AFJ) flow areas must be provided by the user and AF will be calculated automatically by the code.

Inlet flow area (AFI) - inlet (suction) flow area, if not constant.

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Treat H, Q as Coefficients (COEFS) - Option to use coefficient-based performance parameters. If not checked (COEFFS=NO) then dimensional head and flow rates (H and G) are used to define performance parameters. If checked (COEFS=YES), then H is interpreted as head coefficient Ψ instead of head or pressure drop, and G is interpreted as flow coefficient Φ instead of mass or volumetric flow rate. This has an effect not only on the H/Ψ and G/Φ values, but also on HPMP, GPMP, DGDH, and TCF (but *not* SSCF, ANPSH, RNPSH, ANSS, nor RNSS).

H (head) Conversion Factor (HCF) - If Treat H, Q as Coefficients is checked, this is the conversion factor applied to H/SPD^2 to yield Ψ: Ψ = HCF*H/SPD^2. For US Customary units, with H in feet and SPD in rpm, then HCF = g/D^2 where D is the rotor diameter and g is the acceleration of gravity.

Q (flow) Conversion Factor (GFC) - If Treat H, Q as Coefficients is checked, this is the conversion factor applied to G/SPD to yield Φ: Φ= GCF*G/SPD. For US Customary units, with G in gpm and SPD in rpm, then either GCF = 1/(A*D) or perhaps GCF=1/D^3 where D is the rotor diameter and A is the flow area.

Head unit identifier (HU) - Integer flag for head units: see Table 3. These units will apply to H values, but if COEFS=YES then HPMP and all array head values will be nondimensional or at least quasi-nondimensional according to HPMP = Ψ = HCF*H/SPD^2 (where H is given by HU units), and DGDH is in units of GPMP/ HPMP.

TABLE 3. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
0	default: m if uid=si, ft if uid=eng	
1	m	
2	cm	
3	mm	
4	cmH_2O, 4°C	pressure units
5	mmH_2O, 4°C	pressure units
6	mmHg, 0°C (Torr)	pressure units
7	Pa	pressure units
8	kPa	pressure units
9	Bar	pressure units
101	ft	

TABLE 3. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
102	in	
103	inH_20 (39.2°F)	pressure units
104	inHg (32°F)	pressure units
105	psi	pressure units
106	psf	pressure units
201	(absolute) pressure ratio	TABULAR, TURBINE, COMPRESS, or COMPPD only
202	Martin Beta Factor	TABULAR only

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 4. These units will apply to G values, but if COEFS=YES then GPMP and all array flow values will be nondimensional or at least quasi-nondimensional according to GPMP = Φ = GCF*G/SPD (where G is given by GU units).

TABLE 4. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
0	default: m^3/s if uid=si, ft^3/hr if uid=eng	
1	m^3/s	
2	m^3/min	
3	m^3/hr	
4	L/s	
5	L/min	
6	L/hr	
7	cm^3/s	
8	cm^3/min	
9	cm^3/hr	
10	mm^3/s	
11	mm^3/min	
12	mm^3/hr	
13	kg/s	mass flow rate units
14	kg/min	mass flow rate units
15	kg/hr	mass flow rate units
101	ft^3/s (CFS)	
102	ft^3/min (CFM)	
103	ft^3/hr (CFH)	
104	gal/s (US GPS)	
105	gal/min (US GPM)	

TABLE 4. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
106	gal/hr (US GPH)	
107	in^3/s	
108	in^3/min	
109	in^3/hr	
110	lb_m/s	mass flow rate units
111	lb_m/min	mass flow rate units
112	lb_m/hr	mass flow rate units

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then head is defined on the basis of total or stagnation pressures, and either AF or both AFI and AFJ are required inputs. Otherwise (ISTP=NO or equivalently, ISTP=SS), static inlet/outlet pressures are used without accounting for dynamic head. ISTP=TS (total inlet, static outlet) and ISTP=ST (static inlet, total outlet) are also available.

TORQ Correction (TCF) - Correction factor for the PUMP output variable TORQ. The code calculates the hydraulic torque as:

TORQ = TCF*|GPMP|*HPMP/(EFFP*SPD)

where GPMP and HPMP are in units defined by COEFS, GU, and HU, and TCF is input by the user to perform whatever unit conversions are necessary. TORQ is a translatable path output variable. TCF defaults to unity.

In US Customary units, if HPMP is in feet and GPMP is in gpm and SPD is in rpm, then to covert TORQ to lb_f-ft use:

```
TCF = DL#UP*0.1337/(2*pi)      $ DL#UP is the upstream density,
                               $ 0.1337 converts gallon to ft3
                                 $ "pi" is a built-in constant
```

If COEFS=YES in the above case, with GCF=HCF=1.0, then:

```
TCF = DL#UP*0.1337 *SPD#THIS**3 /(2*pi)
```

Library Option - The library option allows the creation of a user-defined component library. The library saves all information in the current form to a database which can then be accessed and used with any model (including those with different units). It is the user's responsibility to define library locations, library names, and to control access to the library. If a library item is updated, the changes will not automatically propagate to a model, rather the user must go into the library dialog and reapply the library item to bring the updates into each device in a model. If sharing a library, be aware that other users can add, redefine, and delete entries in the library. A common library can be used for all Sinaps components or individual libraries can be created, e.g. a pump library, a compressor library, a pipe library, etc.

Note: Library items are unit specific. When a device is saved to the library, the model unit identifier (UID) is saved with the library item. If the library is brought into a model with a different UID value, a warning message will appear.

Select Library - Use this button to browse to, or create, a library file (*.xml format). When creating a new library file, click "Browse" to open browser window, browse to a location for the new library file, and type a name into the "File Name" field, e.g. turbomachine_library or turbomachine_library.xml. This file can be saved locally or on a network server. Once a library has been selected, the path and name of the library will appear in the adjacent field.

Library Selection - This field provides a pull-down list of all components defined in the selected library. If the library is empty, no entries will be listed.

Add to Library - Use the Add to Library button to add the current component description to the library. When prompted for a name, provide a unique name. The current configuration of the component will be saved. Subsequent changes to the component in the model will *not* change the saved library item. To update a component in the library, edit it in the model and then return to the library dialog and select the saved component name from the library and use the Add to Library button to save the updated component to the library under the same name. For a different name, simply provide a new name when prompted with the Component Name dialog and both prior and updated descriptions will be available in the library.

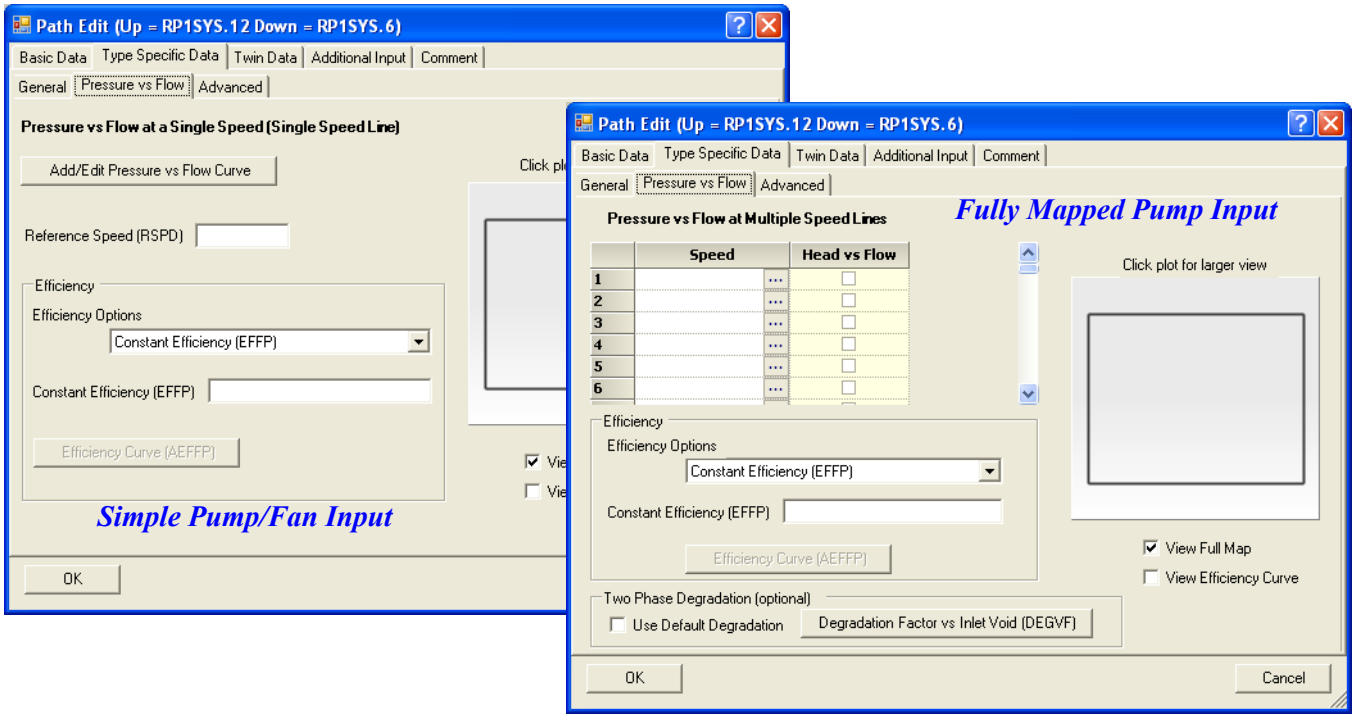
Use Selection - This button populates the component form with information that had been previously saved in the library. After a library has been specified and the Library Selection pull down has be used to select which component is to be imported, use the Use Selection button to import data to the current form. All fields will be imported.

Pump Pressure vs Flow Tab. The contents of the Pressure vs Flow tab will vary depending on which type of pump is being created, a simple pump or a fully mapped pump. In both types of pumps, Sinaps tables are used to input the data. Once input, Sinaps will create the proper arrays to pass into SINDA/FLUINT when running the model. Note that tables created by pumps (and other turbomachinery devices) will not appear in the Sinaps Table Manager since they are only editable through the device dialog, in this case the pump edit form. Performance curves are unique for each device so although two pumps may have identical performance curves, they cannot share a performance curve (updating one will not change the second).

Warning: Tables are used to input turbomachinary data. In general Tables explicitly recognize integer (treated as such in logic) and real numbers. For turbomachinary data inputs, data must be input as real numbers. i.e. you must include the decimal point. Inputting data as an integers will result in an error when executing a case.

Hint: If the model contains multiple pumps all based on the same performance curve(s), rather than creating each pump individually, it is recommended to create one pump, save it to the component library and then import the data from the library into the other pump

FIGURE 5-27. Pump Pressure vs Flow Tab



Simple Pump/Fan Inputs - A simple pump or fan is a device which can be defined by a single pressure vs. flow curve. where the pump input is scalable (similarity rule applies) based on the reference speed (RSPD).

Add/Edit Pressure vs Flow Curve (HG method) - Applicable to a simple pump/fan only. This button will open the table dialog allowing the input of a single pump head-flow curve: head (H: units according to HU) as a function of flow rate (G: units according to GU). G must increase monotonically, and no two adjacent H values may be equal (i.e., the slope dH/dG cannot be zero). If COEFS=YES, the curve will consist of Φ vs Ψ the head and flow rate *coefficients*. When defining the table, data can be real numbers, registers or expressions. Table conversion/scaling options can be used but if COEFS=NO the final units must match the selected values of HU and GU.

Reference pump speed (RSPD) - The reference speed associated with the HG curve, define in consistent user units (same as SPD). This is used to allow scaling based on standard scaling methods. In an ideal pump, "similarity rules" apply: the H scales perfectly with SPD^2 and the G scales linearly with SPD (with other variations fixed, such as pump rotor diameter). The RSPD input is only applicable if a simple pump (HG method) is used and COEFS=NO. If the AEFFP= method is used as well, its G values will be scaled with |SPD/RSPD|.

Fully Mapped Pump Inputs - A full map alternative to the single-curve "HG= method," SPEEDS is the array ID containing a list of speeds for which head and flow rate curves (and possibly efficiencies) will be specified via the HLIST, GLIST, and optionally the ELIST arrays. Like HG, SPEEDS is an array ID, and the entries in that array must be real numbers or expressions in the same units as SPD.

Pressure vx Flow at Multiple Speed Lines - This "Speed" table allows the flow map to be defined. Input the individual speeds in the left column (monotonically increasing as you descend in the table). For each speed input, a head versus flow table must be defined. After inputting a speed value click the adjacent [...] button to access the table input dialog. Note that the format of the Speed table will change depending on various options selected in the form. For example, selecting a full efficiency map (ELIST) in the Efficiency Options will add an efficiency column to the right of the Head vs Flow column. When the head versus flow table input has been completed for each speed the check box in the Head vs Flow column will automatically toggle to a checked state indicating data has been defined. Note the speeds in the speed are used by Sinaps to create the SPEEDS array.

When creating the performance table input at each speed, note the submodel, table ID, and structure are automatically assigned. A name is automatically assigned to the table, however the user can arbitrarily change this value. The performance table for *all speeds* must be the same length. Table conversion/scaling options can be used but the resolved values must be in units consistent with COEFS, HU and GU. From these various table inputs, Sinaps will automatically create the HLIST, GLIST, and individual H and G curves in the proper format to pass into SINDA/FLUINT. If a full efficiency map is defined, the ELIST and associated efficiency curve arrays will also be created.

Efficiency Options - There are multiple options for defining efficiency on a pump, a constant value of efficiency (EFFP), a single curve of efficiency versus flow (AEFFP), or a full efficiency map (only applicable for a fully mapped pump). Pump efficiencies are optional but should be specified or at least estimated. Efficiencies are used to calculate heating and torque.

Constant Efficiency (EFFP) - Pump efficiency as a single user defined value (typical values are between 0.6 and 0.8, for example).

Efficiency Curve (AEFFP) - This button brings up the table dialog in which the submodel, number and structure are automatically defined by Sinaps. The user must input two columns of data defining efficiency (between 0 and 1) and flow G (or flow coefficient Φ if COEFS=YES). End values will be used if the provided range is exceeded; no extrapolations are made for this variable. If RSPD has been specified (above), then G will be scaled with |SPD/RSPD|. Table conversion/scaling options can be used but the resolved values must be in units consistent with COEFS and GU. Sinaps will use the data in the Efficiency Curve table to create the AEFFP array input for SINDA/FLUINT.

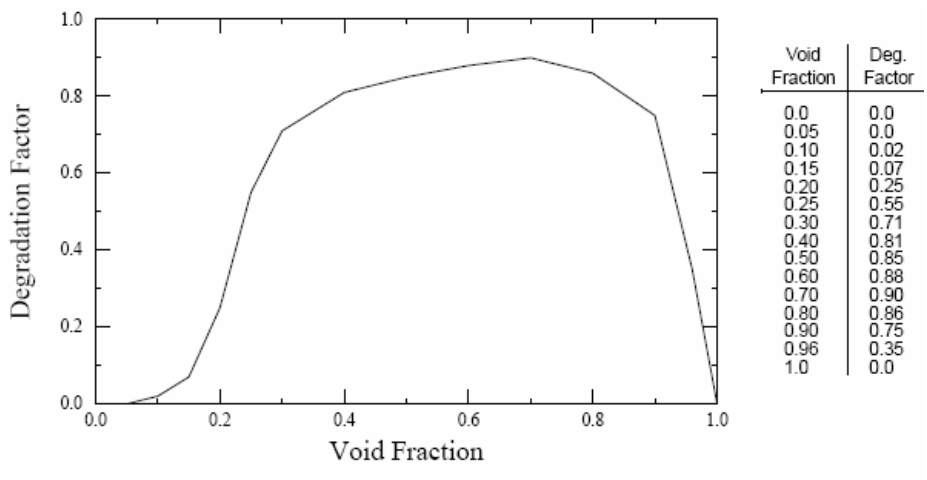
Full Efficiency Map - When a full efficiency map is selected, an efficiency column will be added to the Speed table. To input the efficiency data, click the [...] button next to each speed cell and edit the table to add the pump hydraulic efficiency values (between 0 and 1). Once the data has been defined for each speed, the check box in the Efficiency column of the Speed table will toggle to a checked state indicating content. Sinaps will use the efficiency map data to create the ELIST array and the associated efficiency arrays at each speed.

Data Plots - Whether defining a simple or fully mapped device, the plot area will update automatically to show either a single performance curve or a family of curves (i.e. performance map) along with efficiency curves. Check boxes are provided to visibility toggles for the pressure vs flow plot and for the efficiency plot. Clicking on a plot will open the plot in the EZXY Plotter for a larger view.

Two Phase Degradation Options - Cavitation corrections, if specified, apply when liquid or two-phase flow is present at the inlet (See section 3.5.13.5 of the SINDA/FLUINT User's Manual for detailed information). A separate and inde-

pendent (albeit superimposed) head degradation factor is applied when two-phase flow is encountered at the inlet: $H_{corrected} = H_{uncorrected}*(1\text{-}DEG_{\alpha})$. To disable this correction (perhaps in favor of cavitation corrections), specify a DEGVF array with zero degradation values (two such points are required as a minimum to define a line).

Use Default Degradation - Checking this option will invoke the built-in curve for DEG_{α}, as shown in below. This default curve represents a reasonable average degradation for several pumps reported in the literature and can be used as a first approximation unless specific pump characteristics are known.

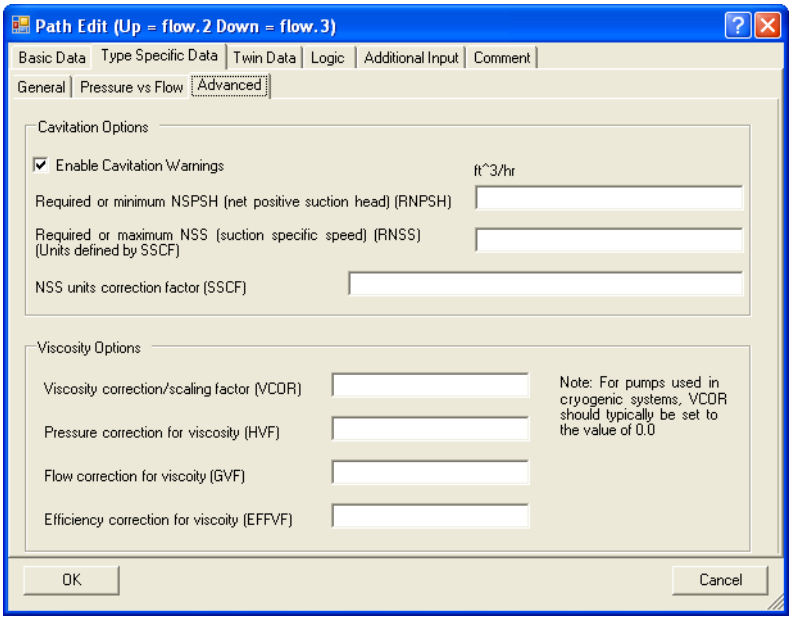


Void Fraction	Deg. Factor
0.0	0.0
0.05	0.0
0.10	0.02
0.15	0.07
0.20	0.25
0.25	0.55
0.30	0.71
0.40	0.81
0.50	0.85
0.60	0.88
0.70	0.90
0.80	0.86
0.90	0.75
0.96	0.35
1.0	0.0

Degradation Factor vs Inlet Void (DEGVF) - This button brings up the table input form with the submodel, number and structure automatically assigned. Input data characterizing the head degradation factor as a function of void fraction at the pump inlet when two-phase is present. To disable the head degradation (perhaps in favor of cavitation corrections), specify a DEGVF table with zero degradation values (two such points are required as a minimum to define a line).

Pump Advanced Tab. The advanced tab provides the ability to set various cavitation and viscosity options.

FIGURE 5-28. Pump Advanced Tab


Path Edit (Up = flow.2 Down = flow.3)
Basic Data
Type Specific Data
Twin Data
Logic
Additional Input
Comment
General
Pressure vs Flow
Advanced
Cavitation Options
Enable Cavitation Warnings
ft^3/hr
Required or minimum NSPSH (net positive suction head) (RNPSH)
Required or maximum NSS (suction specific speed) (RNSS) (Units defined by SSCF)
NSS units correction factor (SSCF)
Viscosity Options
Viscosity correction/scaling factor (VCOR)
Pressure correction for viscosity (HVF)
Flow correction for viscoity (GVF)
Efficiency correction for viscoity (EFFVF)
Note: For pumps used in cryogenic systems, VCOR should typically be set to the value of 0.0
OK
Cancel

Enable Cavitation Warnings - When checked cavitation warnings will be issued based on the following inputs.

Required net positive suction head (RHPSH) - This is the required NPSH in units defined by GU but not by COEFS (it is H not Φ). Can be compared with output variable ANPSH, perhaps using CHKFLASH.

Required suction specific speed (RNSS) - Required Nss (suction specific speed) in units defined by SSCF. Can be compared with output variable ANSS, perhaps using CHKFLASH.

NSS units correction factor (SSCF) - NSS (suction specific speed) units correction factor, such that ANSS= SSCF*SPD*$G^{1/2}$/$ANPSH^{3/4}$ where the flow G and the available NPSH (output variable ANPSH) are defined by GU and HU but *not* by COEFS: they are G and H, not Φ and Ψ.

Viscosity correction/scaling factor (VCOR) - This is the viscosity correction control and scaling factor. If zero, or if neither ELIST nor AEFFP have been input, then no corrections will be performed (HVF=GVF=EFFVF=1.0). If negative, the viscosity degradation will be scaled with |VCOR|. If positive, HVF, GVF, and EFFVF are to be provided by the user.

Pressure correction for viscosity (HVF) - A head correction factor for viscosity (input if VCOR>0, output if VCOR<=0). Apparent head will be divided by this value if the head-flow relationship has been specified using arrays using either the HG= or the HLIST= method.

Flow correction for viscosity (GVF) - A flow correction factor for viscosity (input if VCOR>0, output if VCOR<=0). Flow will be multiplied by this value if the head-flow relationship has been specified using arrays using either the HG= or the HLIST= method.

Efficiency correction factor for viscosity (EFFVF) - An efficiency correction factor for viscosity, applicable if VCOR>0. EFFP will be multiplied by this value if the head-flow relationship has been specified using arrays using either the HG= or the HLIST= method.

5.4.5.10 Turbine Device (TURBINE)

A TURBINE connector is a component-level model of a gas, steam, or hydraulic turbine (whether axial or radial): a turbine whose performance varies as a function of pressure difference or ratio for a given rotational speed. While a TURBINE connector is normally used to represent an entire turbine containing any number of stages, it may also be used to model either a single turbine stage or a radial portion of an axial turbine or stage (e.g., hub, midline, or tip). Such subset models may be combined in series (for single stages) and/or in parallel (for radial portions).

In order to simulate such a turbine or turbine stage, the user must specify mass flow rate (G, whether actual, corrected, or a "flow parameter") as a function of pressure difference or ratio (H) and perhaps rotational speed (SPD). The default H is unitless: a pressure ratio (whether total to total, or total to static).

The flow rate G may be a mass or volumetric flow rate, or it may be specified as a flux (per unit area), or as the flow parameter $GCF*G*T_{in}^{1/2}/P_{in}$ based on the current fluid state at the inlet, or as an effective flow based on a reference inlet state (described later).

A single curve of flow at a single speed may be specified as a function of pressure ratio (by default, although pressure differences may be used as well), or a family of curves can be defined as a function of either actual or equivalent speed. For a family of curves, each flow curve (at each speed) may be a different length if a single set of pressure ratios is used for all such flow curves: it is expected that the curves at low speeds will not extend to the highest pressure ratios.* This flexibility is contrasted with the PUMP and TABULAR options, which require a rectangular grid of M by N points with no "cut off corners." Therefore, the "GLIST" and "HLIST" parameters for a TURBINE and not completely analogous to those of a PUMP or TABULAR.†

The efficiency may be defined as a constant, or as a function of pressure ratio, or as a function of velocity ratios (U/C) and perhaps both velocity ratio and pressure ratio. As an alternative, the power extracted from the flow can be specified as a function of pressure ratio and perhaps speed, with efficiency being calculated as a result.

There are various ways to represent the pressure-flow performance map depending on information available. As will be detailed later, "flow (G)" may actually be a mass flow rate or mass flux, a volumetric flow rate or velocity, a flow parameter corrected for inlet conditions ($T^{1/2}/P$), or a equivalent flow based on a reference state. The choice of G units and method are elected using the unit flag GU combined with the reference method MREF and flow area flag NAF. Similarly, "pressure (H)" may be a head or pressure difference (for a hydraulic turbine perhaps), or a pressure ratio according to the unit flag HU, and these pressures may be optionally chosen to be based on static or total conditions using the ISTP control.

The choices for specifying the pressure-flow relationships include:

1. A single curve of flow versus pressure difference or pressure ratio ("the GH= method").

* This subject is related to choking, which is assumed in a TURBINE device to be contained within the performance curves.

† Indeed, the GLIST and HLIST for a PUMP are not completely analogous to that of a TABULAR connector. The actual means of specifying G and H varies from device to device despite the commonality of input keywords.

2. A set of curves, each at increasing rotational speed. (If an equivalent flow method is used, this "speed" may optionally be chosen to be an equivalent speed based on a reference state.) This method will be referred to as "the GLIST= method." There are two submethods available for the GLIST= method, depending on whether a single set of pressure ratio points is used for all speed curves, or whether a *family* of such H curves is supplied, one for each speed.
    a. In the "HLIST= submethod," a list of pressure arrays is input, with each value corresponding to the same location in the corresponding GLIST array. All of the arrays within both HLIST and GLIST must then be the same size: the same number of data points must be provided for each speed line. HLIST and GLIST then both becomes a "list of lists," and the map space is rectangular (N speeds by M data points at each speed) with no cut-off corners.
    b. In the contrasting "HLIST1= submethod," a single array of pressure points can be used to refer to all of the speed lines: HLIST1 is a single list, not a "list of lists" as is HLIST. Using this submethod, the points within the flow arrays (GLIST) must be listed at the same pressure points. For example, the first flow values in each GLIST array should correspond to the first pressure value in the HLIST1 array. To accommodate the missing data for low speeds and high pressure ratios, the GLIST arrays need not all be the same length: the points at higher pressure pressures may be omitted if unavailable: the corner may be cut off.
3. If the pressure-flow maps exist as functions or are provided via calls to (perhaps COM-linked external) turbine design codes, then the current value of H and G can provided directly as G(H): flow as a function of current pressure difference or ratio. Furthermore, SINDA/FLUINT requires the slope at the current speed and pressure ratio: dG/dH, the inverse of which (dH/dG) cannot be zero. In other words, the user can omit performance arrays and can instead manipulate the turbine parameters directly in logic and/or expressions. The variables for H and G are HTURB and GTURB. The code provides the current value of HTURB (head, pressure drop, or pressure ratio), and the user must supply the corresponding values for GTURB (flow rate, mass flux, flow parameter, equivalent flow, etc.) and DGDH (the slope). This method will be referred to as "the GTURB= method."

Turbine isentropic efficiencies are optional but should be specified or at least estimated (typical values are between 0.3 and 0.9, for example). Efficiencies are used to calculate power extracted and torque, and may be specified as either total-total or total-static (matching the pressure method described above via the ISTP parameter).

There are five ways to specify efficiency, η, or alternatively, the extracted power:

1. as a single value, EFFP (updated in logic or expressions). This method will be referred to as "the EFFP= method."
2. as an interpolated array of efficiency vs. velocity ratio (U/C). In this case, U is the tangential blade tip velocity, which requires DTURB (the turbine diameter) to be specified as an input. C is the isentropic spouting velocity for a gas or steam turbine, calculated assuming an isentropic expansion from a total inlet state to a static outlet state (independent of ISTP). For hydraulic turbines, C is the fluid jet speed, and is calculated based on the Bernoulli equation again assuming a total inlet state and a static outlet state. This method will be referred to as "the AEFFP= method."
3. as an interpolated series of arrays of efficiency vs. velocity ratio (U/C), with different curves supplied for each of a series of pressure differences or ratios (corresponding to the same set of pressures used to define the pressure-flow relationship, perhaps, if the GLIST= method were employed). This method will be referred to as "the ELISTR= method." The ELISTR= method requires that the corresponding pressure ratios be supplied as HLIST1 even if pressure ratios are supplied using the HLIST= submethod.

4. if a full map of G-H has been provided via the GLIST= method, then a full map of efficiencies can be provided in parallel (e.g, full maps of η at the same operating points as the head-flow relationships). This method will be referred to as "the ELIST= method."
5. if a full map of G-H has been provided via the GLIST= method, then a full map of extracted power (W or BTU/hr, depending on UID) can be provided in parallel (e.g, full maps of power at the same operating points as the head-flow relationships). In this case, EFFP becomes an output variable. This method will be referred to as "the PLIST= method."

The code calculates the hydraulic torque as:

```
TORQ      = TCF*QTMK/SPD
```

where TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity.

TORQ is a translatable path output variable. In SI units, the TCF contains the conversion of rpm (assuming those are the units of SPD) to rad/sec, resulting in TORQ units of N-m:

```
TCF =60./(2.*pi)              $ "pi" is a built-in constant
```

In US Customary units, QTMK is in BTU/hr, so assuming SPD is in RPM, to get TORQ in lb_f-ft use:

```
TCF = 778./(2.*pi*60.)        $ 778 converts BTU/hr to lbf-ft/hr
```

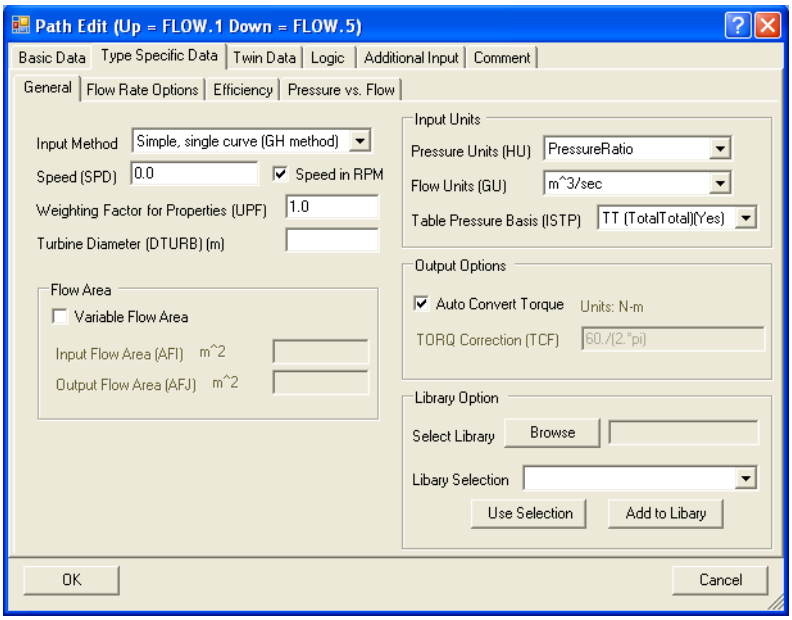
Hydraulic torque may be used along with co-solved ordinary differential equations (e.g., DIFFEQ1) to calculate the current speed of the turbine during transients, or may be used to solve a torque balance equation for steady-states (e.g., for turbopumps or turbochargers). Note that TORQ is positive for energy into the fluid system. It is therefore usually negative for turbines, and so the negative of TORQ should be set proportional to $d\omega/dt$, the rate of change of shaft speed.

Figure 5-29 shows the type-specific edit form for a turbine. Descriptions of each input are described below. For detailed information on describing turbines, choking detection and simulation, torque, heating, and other advanced options, please refer to section 3.5.15 of the SINDA/FLUINT User's Manual.

> Note: The Basic Data tab of the Turbine form has an option to "Use 5.3 Type Specific Data". The Turbine form has been redesigned in version 5.4 to provide increased "user friendliness", assisting the user in determining which inputs are required and providing access to Sinaps Tables for array data input. When a 5.3 or older model is opened, the "Use 5.3 Type Specific Data" option will be checked and the Type Specific Data tab content will appear has the heritage style input form. The user may continue using the heritage form however it is recommended to update to the new form by unchecking the box. Be aware that when updating to the new form, some data (such as performance curves) will need to be reinput. For information on the heritage style form please see "Heritage Turbine Device Form" on page A-6.

Turbine General Tab. Data input on the General Tab applies to all turbines regardless of the input method. For additional information on compressors, see "Turbine Device (TURBINE)" on page 5-68.

FIGURE 5-29. Turbine General Form



Input Method - A turbine can be defined as either a “Simple, single curve (HG method)” using a single turbine flow vs pressure difference (or ratio) curve, or a “Fully Mapping (GLIST method)” which uses a family of curves to define the flow versus pressure difference at various rotational speeds. The input method selection controls the content of the Pressure vs Flow tab allowing only applicable data fields to be displayed.

Current/Initial Speed (SPD) - Current/initial turbine speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency. Note that SPD is in actual units (e.g., rpm) and is not dependent on the value of NSPD. NOTE: To use the ELISTR or AEFFP options, or for the output variable UCR (velocity ratio U/C) to be valid, SPD *must* be in units of rpm.

Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density.

Turbine Diameter (DTURB) - Turbine diameter (in feet or meters, depending on UID). This input is required if either the AEFFP= or the ELISTR= method is used to define efficiency based on velocity ratio (U/C). If DTURB is input, SPD should be in units of RPM such that the output variable UCR (the velocity ratio U/C) can be used, whether or not AEFFP or ELISTR is used.

Variable Flow Area - By default the flow area is defined as a constant area as defined by the field Flow Area (AF) on the General tab of the form. Checking the Variable Flow Area box enables a variable flow area across the path. When checked both the inlet (AFI) and outlet (AFJ) flow areas must be provided by the user and AF will be calculated automatically by the code.

> *Inlet flow area (AFI)* - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Head unit identifier (HU) - Integer flag for head units: see Table 5. These units will apply to H values.

TABLE 5. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
0	default: m if uid=si, ft if uid=eng	
1	m	
2	cm	
3	mm	
4	cmH_2O, 4°C	pressure units
5	mmH_2O, 4°C	pressure units
6	mmHg, 0°C (Torr)	pressure units
7	Pa	pressure units
8	kPa	pressure units
9	Bar	pressure units
101	ft	
102	in	
103	inH_20 (39.2°F)	pressure units
104	inHg (32°F)	pressure units
105	psi	pressure units
106	psf	pressure units
201	(absolute) pressure ratio	TABULAR, TURBINE, COMPRESS, or COMPPD only
202	Martin Beta Factor	TABULAR only

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 6. These units will apply to G values. See also MREF and NAF for further modifiers of these basic units.

TABLE 6. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
0	default: m^3/s if uid=si, ft^3/hr if uid=eng	
1	m^3/s	
2	m^3/min	
3	m^3/hr	
4	L/s	
5	L/min	
6	L/hr	
7	cm^3/s	

TABLE 6. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
8	cm^3/min	
9	cm^3/hr	
10	mm^3/s	
11	mm^3/min	
12	mm^3/hr	
13	kg/s	mass flow rate units
14	kg/min	mass flow rate units
15	kg/hr	mass flow rate units
101	ft^3/s (CFS)	
102	ft^3/min (CFM)	
103	ft^3/hr (CFH)	
104	gal/s (US GPS)	
105	gal/min (US GPM)	
106	gal/hr (US GPH)	
107	in^3/s	
108	in^3/min	
109	in^3/hr	
110	lb_m/s	mass flow rate units
111	lb_m/min	mass flow rate units
112	lb_m/hr	mass flow rate units

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then pressures and other state variables are defined on the basis of total or stagnation conditions, and either AF or both AFI and AFJ are required inputs. By default ISTP=TT, and so both inlet and outlet states are based on total (stagnation) conditions and therefore flow areas (AF or both AFI and AFJ) are required. ISTP=TS (total-static) and ISTP=ST (static-total) are also available as options. ISTP also affects the meaning of TREF and PREF.

Auto Convert Torque - Checkbox to invoke automatic torque conversion (applicable only if the Speed in RPM box is checked). User may uncheck this box to activate the TCF field and assume control of the torque conversion factor. If checked, the TCF field will gray out and be defined by Sinaps based on the model units (UID). If using SI units, the TCF field will contain the conversion of rpm to rad/sec, resulting in TORQ units of N-m:

```
TCF = 60./(2.*pi)
```

In US Customary units, torque will be converted into $lb_f ft$ using:

```
TCF = 778./(2.*pi*60.)
```

TORQ Correction (TCF) - Correction factor for the output variable TORQ. The code calculates the torque as:

```
TORQ = TCF*QTMK/SPD
```

where TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity. TORQ is a translatable path output variable.

Library Option - The library option allows the creation of a user-defined component library. The library saves all information in the current form to a database which can then be accessed and used with any model (including those with different units). It is the user's responsibility to define library locations, library names, and to control access to the library. If a library item is updated, the changes will not automatically propagate to a model, rather the user must go into the library dialog and reapply the library item to bring the updates into each device in a model. If sharing a library, be aware that other users can add, redefine, and delete entries in the library. A common library can be used for all Sinaps components or individual libraries can be created, e.g. a pump library, a compressor library, a pipe library, etc.

Select Library - Use this button to browse to, or create, a library file (*.xml format). When creating a new library file, click "Browse" to open browser window, browse to a location for the new library file, and type a name into the "File Name" field, e.g. turbomachine_library or turbomachine_library.xml. This file can be saved locally or on a network server. Once a library has been selected, the path and name of the library will appear in the adjacent field.

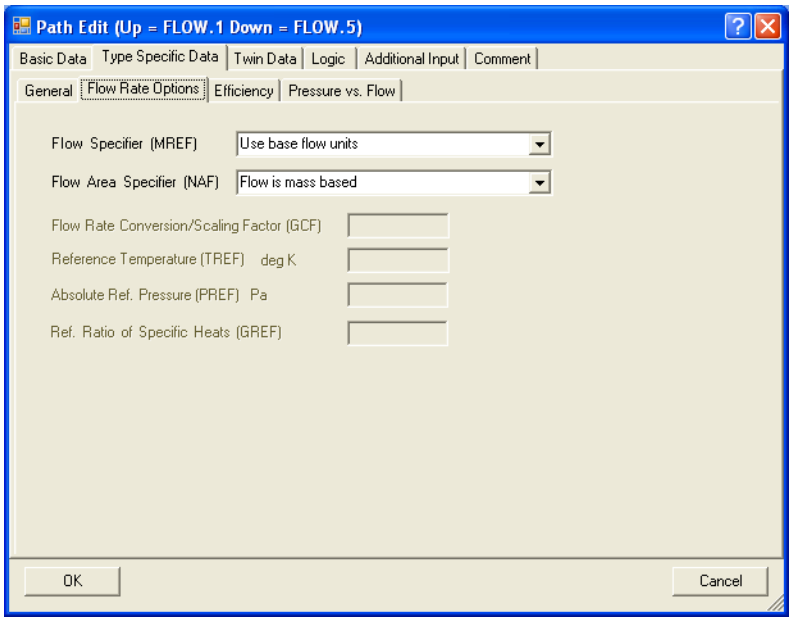
Library Selection - This field provides a pull-down list of all components defined in the selected library. If the library is empty, no entries will be listed.

Add to Library - Use the Add to Library button to add the current component description to the library. When prompted for a name, provide a unique name. The current configuration of the component will be saved. Subsequent changes to the component in the model will *not* change the saved library item. To update a component in the library, edit it in the model and then return to the library dialog and select the saved component name from the library and use the Add to Library button to save the updated component to the library under the same name. For a different name, simply provide a new name when prompted with the Component Name dialog and both prior and updated descriptions will be available in the library.

Use Selection - This button populates the component form with information that had been previously saved in the library. After a library has been specified and the Library Selection pull down has be used to select which component is to be imported, use the Use Selection button to import data to the current form. All fields will be imported.

Turbine Flow Rate Options. The base units for the flow rate, G, are defined using the GU parameter whose meaning is summarized in Table 6 on page 72. However, the actual flow rate value provided in the tables, G_{table}, may deviate from the units specified by GU. The Flow Rate Options form (Figure 5-30 on page 75) provides access for specifying how flow rate is being defined if not based on the units specified by GU.

FIGURE 5-30. Turbine Flow Rate Options Form



Flow Specifier (MREF) - Integer mass flow reference flag. By default MREF is set to base flow units or zero, and hence G values provided are assumed to be based on the units specified by GU.

*Mass corrected (GCF*G*T**0.5/P)*, or MREF=1, instead means that the G values are actually corrected mass flows: the values supplied as "G" are actually to be interpreted as $G_{table} = GCF*G_{GU}*T^{1/2}/P$ where T and P are the absolute inlet temperature and pressure, respectively (static or total, depending on ISTP), and G_{GU} is determined by the unit selector GU. Absolute means the units of T are either K or R (depending on UID), and the units of P are either psia or absolute Pascal. GCF may be used for unit conversions or scaling.

*Mass corrected (GCF*G*(T/Tref)**0.5/(P/Pref))*, or MREF=2, $G_{table} = GCF*G_{GU}*(T/T_{ref})^{1/2}/(P/P_{ref})$ where the reference temperature and pressure (TREF, PREF) are specified separately.

Mass corrected function of Tref, Pref, Gref, or MREF=3, a more complicated correction is made for equivalent conditions (see main text) using a reference temperature and pressure and also a reference ratio of specific heats, γ_{ref} (GREF).

*Mass corrected (GCF*G*(RT/g)**0.5/P)*, MREF=4 invokes a slightly more complicated variation of MREF=1 that includes gas properties and unit conversions: $G_{table} = GCF*G_{GU}* (RT/g_c)^{1/2}/P$ where R is the gas constant, and g_c=1 if UID=SI or g_c=32.174 if UID=ENG. MREF may be used in combination with NAF.

(Note "G_{table}" is equivalent to "GCOMP" when using the GCOMP= method.)

Flow Area Specifier (NAF) - Integer mass flux basis flag. By default flow is mass based (NAF=0), and hence G values provided are assumed to be based on the units specified by GU.

Mass flux based on average area (NAF=1) instead means that the G values are actually mass fluxes based on the flow area AF, with the parameter GCF available for conversions or scaling. In other words, for NAF=1 the values supplied as "G" are actually to be interpreted as $G_{table} = GCF*G_{GU}/AF$ where G_{GU} is determined by the unit selector GU.

Mass flux based on upstream area (NAF=2), Gtable = GCF*GGU/AFI

Mass flux based on downstream area (NAF=3), Gtable = GCF*GGU/AFJ

(Note "G_{table}" is equivalent to "GCOMP" when using the GCOMP= method.)

Conversion factor applied to flow rate (GFC) - A G conversion factor (defaulting to unity) that is active if either NAF or MREF is nonzero. *GCF is ignored if MREF=NAF=0.*

Reference Temperature (TREF) - The reference temperature (K or R, depending on UID, and static or total, depending on ISTP) used for MREF=2 or MREF=3.

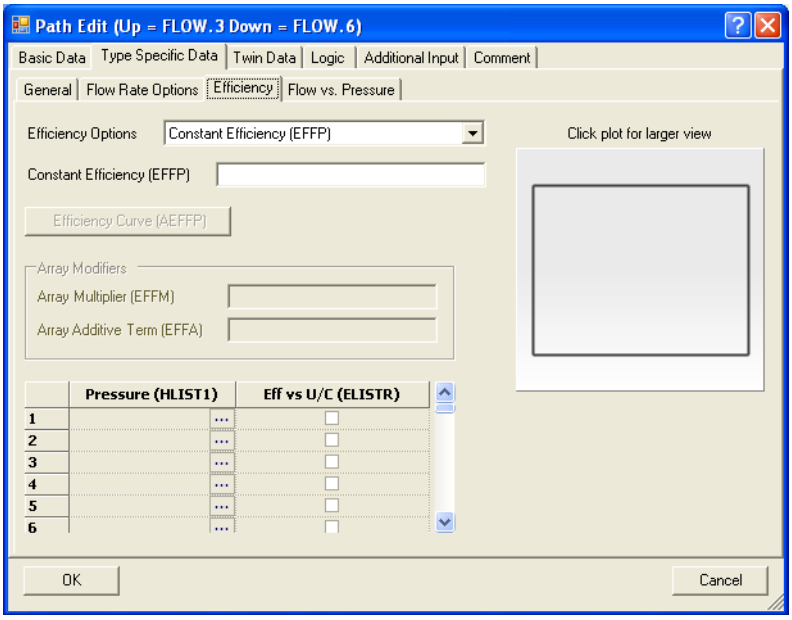
Absolute Reference Pressure (PREF) - The reference pressure (absolute Pascal or psia, depending on UID, and static or total, depending on ISTP) used for MREF=2 or MREF=3.

Reference Ratio of Specific Heats (GREF) - The reference ratio of specific heats, $\gamma_{ref} = Cp/Cv$, used for MREF=3.

Turbine Efficiency Options. Turbine isentropic efficiencies are optional but should be specified or at least estimated (typical values are between 0.3 and 0.9, for example). Efficiencies are used to calculate power extracted and torque, and may be specified as either total-total or total-static (matching the pressure method described via the ISTP parameter).

Warning: Tables are used to input turbomachinary data. In general Tables explicitly recognize integer (treated as such in logic) and real numbers. For turbomachinary data inputs, data must be input as real numbers. i.e. you must include the decimal point. Inputting data as an integers will result in an error when executing a case.

FIGURE 5-31. Turbine Efficiency Options Form



Efficiency Options - There are multiple options for defining efficiency, a constant value of efficiency (EFFP), a single curve of efficiency versus flow (AEFFP), or a full efficiency map (only applicable for the full map option). Efficiencies are optional but should be specified or at least estimated. Efficiencies are used to calculate heating and torque. Alternatively in lieu of efficiency, the added or absorbed power can be mapped.

Constant Efficiency (EFFP) - Efficiency defined as a single user defined value which can be updated in logic or expressions (typical values are between 0.6 and 0.8, for example).

Single Curve of Efficiency vs Velocity Ratio (AEFFP) - Define a table of efficiency vs. flow velocity ratio (U/C). In this case, where U is the tangential tip velocity (based on SPD and DTURB) and C is the isentropic spouting velocity for a gas turbine (fluid jet speed for a hydraulic turbine). This option is only applicable if the speed is in RPM (Speed in RPM box checked). Click the Efficiency Curve button to bring up the table dialog in which the submodel, number and structure are automatically defined by Sinaps. The user must input two columns of data defining the velocity ration (U/C) and efficiency (between 0 and 1). End values will be used if the provided range is exceeded; no extrapolations are made for this variable. Table conversion/scaling options can be used but the resolved values must be in units consistent with GU. Sinaps will use the data in the Efficiency Curve table to create the AEFFP array input for SINDA/FLUINT.

Full Efficiency Map - IF the turbine is defined using a full performance map, the efficiency map option can be selected to define a corresponding map of isentropic efficiencies. When selected, an efficiency column will be added to the Speed table on the Flow vs Pressure tab. To input the efficiency data, click the [...] button next to each speed cell and edit the table to add the efficiency values (between 0 and 1). Once the data has been defined for each speed, the check box in the Efficiency column of the Speed table will toggle to a checked state indicating content. Sinaps will use the efficiency map data to create the ELIST array and the associated efficiency arrays at each speed.

Map of Extracted Power (PLIST) - If a full performance map has been defined, instead of efficiencies, a map of extracted power (W or BTU/hr, depending on UID) can be provided. In this case efficiency (EFFP) becomes an ouput variable. When selected, a Power column will be added to the Speed table on the Flow vs Pressure tab. To input the power data, click the [...] button next to each speed cell and edit the table to define the extracted power values. Once the data has been defined for each speed, the check box in the Power column of the Speed table will toggle to a checked state indicating content. Sinaps will use this data to create the PLIST array and the associated power arrays for each speed.

Efficiency vs Velocity Ratio Interpolated Against Pressure (ELISTR) - If a full performance map has been defined, efficiency can be defined using an interpolated series of arrays of efficiency vs. velocity ratio (U/C), with different curves supplied for each of a series of pressure differences or ratios. This option is only applicable if the speed is in RPM (Speed in RPM box checked). When this option is selected, the Pressure vs ELISTR table on the efficiency tab will become active. If Sinaps will use this data to create the PLIST array and the associated power arrays for each speed. If a common pressure array has been defined on the Flow vs Pressure tab, the pressures will already be displayed in the table, if not, pressures must be defined. For each pressure in the table click the [...] button next to the pressure cell and edit the table to define the velocity ration in column 1 and efficiency in column 2. Once the data has been defined for each pressure, the check box in the Eff vs U/C column of the table will toggle to a checked state indicating content. Sinaps will use this data to create the ELISTR array and its associated efficiency vs U/C arrays for each pressure. It will also create the HLIST1 array.

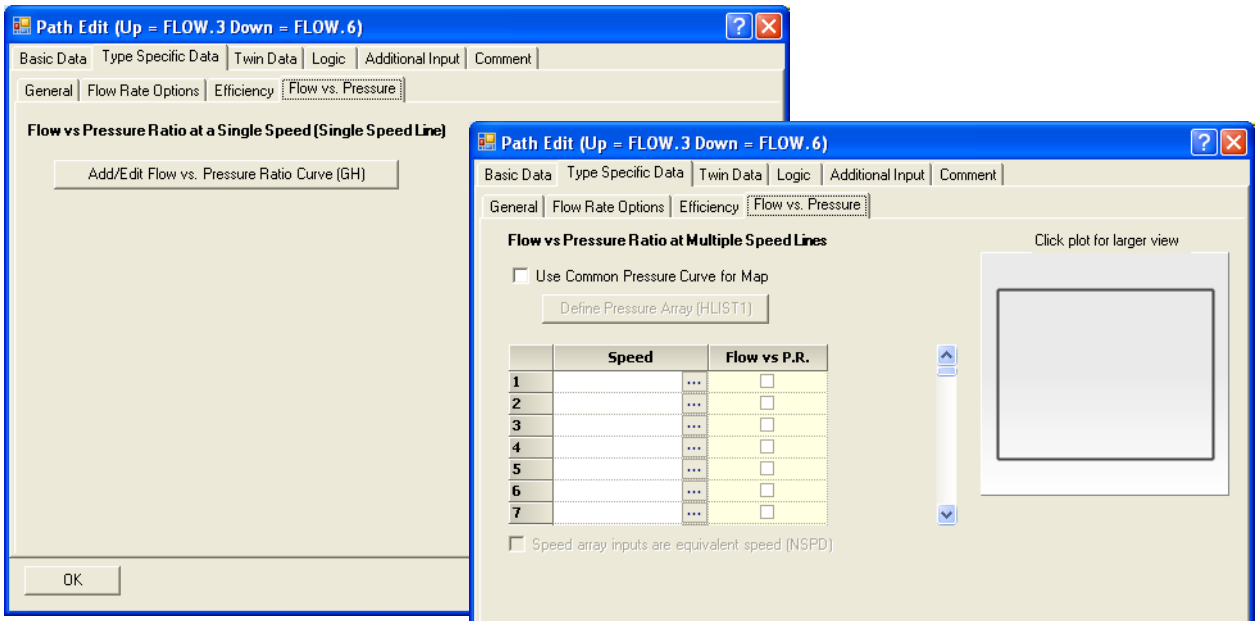
Array Multiplier (EFFM) - Multiplier to use for EFFP values extracted from array options (ELIST, AEFFP, or PLIST).

Array Additive Term (EFFA) - Additive term to use for EFFP values extracted from array options (ELIST, AEFFP, or PLIST).

Turbine Flow vs Pressure Options. There are various ways to represent the pressure-flow performance map depending on information available. The “flow (G)” may actually be a mass flow rate or mass flux, a volumetric flow rate or velocity, a flow parameter corrected for inlet conditions ($T^{1/2}/P$), or a equivalent flow based on a reference state. The choice of G units and method are elected using the unit flag GU (set on the General tab) combined with the reference method MREF and flow area flag NAF (both defined on the Flow Rate Options tab). Similarly, “pressure (H)” may be a head or pressure difference (for a hydraulic turbine perhaps), or a pressure ratio according to the unit flag HU (set on the General tab), and these pressures may be optionally chosen to be based on static or total conditions using the ISTP control (set on the General tab).

Warning: Tables are used to input turbomachinary data. In general Tables explicitly recognize integer (treated as such in logic) and real numbers. For turbomachinary data inputs, data must be input as real numbers. i.e. you must include the decimal point. Inputting data as an integers will result in an error when executing a case.

FIGURE 5-32. Turbine Flow vs Pressure Form



Simple, Single Curve Input - A simple compressor is a device which can be defined by a single pressure vs. flow curve.

Add/Edit Flow vs Pressure Ration Curve (GH method) - Define a table of turbine flow rate (units according to GU) as a function of pressure ratio or difference (units according to HU). The pressure ratio/difference must increase monotonically. No two adjacent pressure values may be equal. Extrapolations will be flagged as non-zero LIMP values.

Fully Mapped Turbine Inputs - A full map alternative to the single-curve, this is referred to as the GLIST method. The performance is defined by a series of flow rate vs pressure ratio or difference curves for a family of speeds.

Use Common Pressure Curve for Map (HLIST1) - Checking this box indicates all flow vs pressure curves defined in the performance map will use the same pressure curve. If checked, click the Define Pressure Curve button to define the pressure values. These values will automatically be used in all tables accessed through the speed table. Note that if the efficiency has been defined using the Efficiency vs Velocity Ratio Interpolated Against Pressure option this pressure curve will synchronize to the save pressure curve defined on the Efficiency tab.

Flow vs Pressure at Multiple Speed Lines - The “Speed” table allows a flow verses pressure map to be defined. Input the individual speeds in the left column (monotonically increasing as you descend in the table). For each speed input, a flow versus pressure ratio table must be defined. The performance tables for *all speeds* must be the same length. The pressure ratios in each table must increase monotonically. The speeds in the speed table are used by Sinaps to create the SPEEDS array.

After inputting a speed value click the adjacent [...] button to access the table input dialog. Note that the format of the Speed table will change depending on various options selected in the form. For example, selecting a full efficiency map (ELIST) in the Efficiency Options will add an efficiency column to the right of the Flow vs P.R.

column. When the table input has been completed for each speed the check box in the Flow vs P.R. column will automatically toggle to a checked state indicating data has been defined.

When creating the performance table input at each speed, note the submodel, table ID, and structure are automatically assigned. A name is automatically assigned to the table, however the user can arbitrarily change this value. Table conversion/scaling options can be used but the resolved values must be in units consistent with COEFS, HU and GU. From these various table inputs, Sinaps will automatically create the HLIST, GLIST, and individual H and G curves in the proper format to pass into SINDA/FLUINT. If a full efficiency map is defined, the ELIST and associated efficiency curve arrays will also be created.

5.4.5.11 Variable Displacement Compressor Device (COMPRESS)

A COMPRESS connector is a component-level model of an axial or radial compressor: a compressor whose performance varies as a function of flow rate or pressure difference ratio for a given rotational speed. While a COMPRESS connector is normally used to represent an entire compressor containing any number of stages, it may also be used to model either a single compressor stage or a radial portion of an axial compressor or stage (e.g., hub, midline, or tip). Such subset models may be combined in series (for single stages) and/or in parallel (for radial portions). For positive displacement compressors (e.g., scroll, vane, or piston), see COMPPD in Section 5.4.5.12.

In order to simulate such a compressor or compressor stage, the user must specify pressure difference or ratio (H) as a function of mass flow rate (G, whether actual, corrected, or a “flow parameter”) and perhaps rotational speed (SPD). The default H is unitless: a pressure ratio (whether total to total, or total to static).

The flow rate G may be a mass or volumetric flow rate, or it may be specified as a flux (per unit area), or as the flow parameter $GCF*G*T_{in}^{1/2}/P_{in}$ based on the current fluid state at the inlet, or as an effective flow based on a reference inlet state (described later).

A single curve of pressure ratio at a single speed may be specified as a function of flow, or a family of curves can be defined as a function of either actual or equivalent speed

The isentropic efficiency may be defined as a constant, or as a function of flow, or as a function of both speed and flow. As an alternative, the power added to the flow can be specified

There are various ways to represent the pressure-flow performance map depending on information available. As will be detailed later, “flow (G)” may actually be a mass flow rate or mass flux, a volumetric flow rate or velocity, a flow parameter corrected for inlet conditions ($T^{1/2}/P$), or a equivalent flow based on a reference state. The choice of G units and method are elected using the unit flag GU combined with the reference method MREF and flow area flag NAF. Similarly, “pressure (H)” may be a head or pressure difference, or a pressure ratio according to the unit flag HU, and these pressures may be optionally chosen to be based on static or total conditions using the ISTP control.

The choices for specifying the pressure-flow relationships include:

1. A single curve of pressure difference or pressure ratio versus flow rate. The pressure ratios need not decrease monotonically, but the flow values must increase monotonically. This method will be referred to as “the HG= method.” The initial point is assumed to represent the surge or stall line, and the last point is assumed to represent the choke line.

2. A set of curves, each at increasing rotational speed. (If an equivalent flow method is used, this "speed" may optionally be chosen to be an equivalent speed based on a reference state. Also, if data is supplied along "R-lines" it can be transposed into equivalent data along speed lines as described later.) The pressure ratios need not decrease monotonically, but the flow values must increase monotonically. This method will be referred to as "the HLIST= method." The initial points at each speed are assumed to represent the surge or stall line, and the last points at each speed are assumed to represent the choke line. However, the surge line may be specified independently using the GSURGE or HSURGE methods.
3. If the pressure-flow maps exist as functions or are provided via calls to (perhaps COM-linked external) compressor design codes, then the current value of H and G can provided directly as G(H): flow as a function of current pressure ratio. Although data is often perceived as H(G) ... pressure ratio as a function of current flow ... SINDA/FLUINT actually requires the reverse: G(H): flow as a function of current pressure ratio. Furthermore, SINDA/FLUINT requires the slope at the current speed and pressure ratio: dG/dH, the inverse of which (dH/dG) cannot be zero. In other words, the user can omit performance arrays and can instead manipulate the compressor parameters directly in logic and/or expressions. The variables for H and G are HCOMP and GCOMP. The code provides the current value of HCOMP (head, pressure drop, or pressure ratio), and the user must supply the corresponding values for GCOMP (flow rate, mass flux, flow parameter, equivalent flow, etc.) and DGDH (the slope). This method will be referred to as "the GCOMP= method."

Compressor isentropic efficiencies are optional but should be specified or at least estimated (typical values are between 0.5 and 0.8, for example). Efficiencies are used to calculate power added and torque, and may be specified as either total-total or total-static (matching the pressure method described above via the ISTP parameter).

There are four ways to specify isentropic (or "power") efficiency, η, or alternatively, the extracted power:

1. as a single value, EFFP (updated in logic or expressions) This method will be referred to as "the EFFP= method."
2. as an interpolated array of isentropic efficiency vs. flow rate (G). This method will be referred to as "the AEFFP= method."
3. if a full map of H-G has been provided via the HLIST= method, then a full map of isentropic efficiencies can be provided in parallel (e.g, full maps of η at the same operating points as the head-flow relationships). This method will be referred to as "the ELIST= method."
4. if a full map of H-G has been provided via the HLIST= method, then a full map of added or absorbed power (W or BTU/hr, depending on UID) can be provided in parallel (e.g, full maps of power at the same operating points as the head-flow relationships). In this case, EFFP becomes an output variable. This method will be referred to as "the PLIST= method."

The code calculates the hydraulic torque as:

```
TORQ      = TCF*QTMK/SPD
```

where TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity.

TORQ is a translatable path output variable. In SI units, the TCF contains the conversion of rpm (assuming those are the units of SPD) to rad/sec, resulting in TORQ units of N-m:

```
TCF = 60./(2.*pi)                $ "pi" is a built-in constant
```

In US Customary units, QTMK is in BTU/hr, so assuming SPD is in RPM, to get TORQ in lb_f-ft use:

```
TCF = 778./(2.*pi*60.)            $ 778 converts BTU/hr to lbf-ft/hr
```

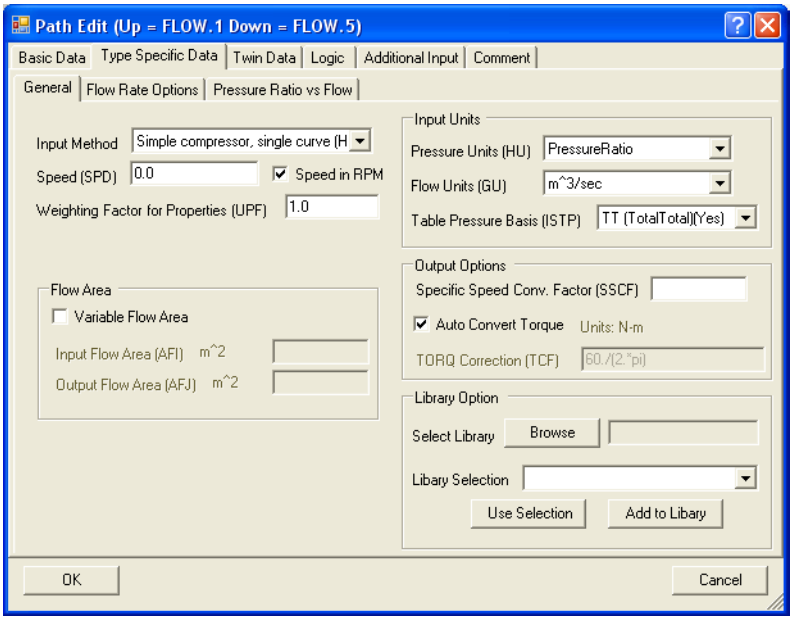
Hydraulic torque may be used along with co-solved ordinary differential equations (e.g., DIFFEQ1) to calculate the current speed of the compressor during transients, or may be used to solve a torque balance equation for steady-states (e.g., for turbobrayton cycles or turbochargers). Note that TORQ is positive for energy into the fluid system. It is therefore usually positive for compressors, and the negative of TORQ (usually a positive number) should be set proportional to $d\omega/dt$, the rate of change of shaft speed.

Figure 5-33 shows the type-specific edit form for a COMPRESS device. This form has multiple tabs for defining general parameters, flow rate options and pressure vs flow options. For each tab, descriptions of each data field are provided below. For detailed information describing compressor usage, surge detection and simulation, torque, heating, and other advanced options, please refer to section 3.5.16 of the SINDA/FLUINT User's Manual.

> Note: The Basic Data tab has an option to "Use 5.3 Type Specific Data". The COMPRESS form has been redesigned in version 5.4 to provide increased "user friendliness", assisting the user in determining which inputs are required and providing access to Sinaps Tables for array data input. When a 5.3 or older model is opened, the "Use 5.3 Type Specific Data" option will be checked and the Type Specific Data tab content will appear has the heritage style input form. The user may continue using the heritage form however it is recommended to update to the new form by unchecking the box. Be aware that when updating to the new form, some data (such as performance curves) will need to be reinput. For information on the heritage style form please see "Heritage Variable Displacement Compressor Form" on page A-8.

Compressor General Tab. Data input on the General Tab applies to all compressors regardless of the input method. For additional information on compressors, see "Variable Displacement Compressor Device (COMPRESS)" on page 5-80.

FIGURE 5-33. Compressor Type Specific Data Form



Input Method - A compressor can be defined as either a “Simple, single curve (HG method)” using a single curve to defined pressure vs flow, or a “Fully Mapped Compressor” which uses a family of curves to define the pressure vs flow at various speeds. The input method selection controls the content of the Pressure vs Flow tab allowing only applicable data fields to be displayed.

Current/Initial Speed (SPD) - Current/initial compressor speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency. Note that SPD is in actual units (e.g., rpm) and is not dependent on the value of NSPD, unlike the values in the SPEEDS array. SPD should be specified even if using the GCOMP= method, since SPD is used for TORQ calculations.

Speed in RPM - This check box allows automatic conversion of output torque. If the current/initial speed is not in units of RPM, uncheck this box. By default this box is checked since the speed units are typically input as RPM. When checked, the Auto Convert Torque box is activated, invoking automatic conversions to be applied to the output torque based on the model units (UID).

Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density.

Variable Flow Area - By default the flow area is defined as a constant area as defined by the field Flow Area (AF) on the General tab of the form. Checking the Variable Flow Area box enables a variable flow area across the path. When checked both the inlet (AFI) and outlet (AFJ) flow areas must be provided by the user and AF will be calculated automatically by the code.

Inlet flow area (AFI) - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Head unit identifier (HU) - Integer flag for head units: see Table 7. These units will apply to H values.

TABLE 7. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
0	default: m if uid=si, ft if uid=eng	
1	m	
2	cm	
3	mm	
4	cmH_2O, 4°C	pressure units
5	mmH_2O, 4°C	pressure units
6	mmHg, 0°C (Torr)	pressure units
7	Pa	pressure units
8	kPa	pressure units
9	Bar	pressure units
101	ft	
102	in	
103	inH_20 (39.2°F)	pressure units
104	inHg (32°F)	pressure units
105	psi	pressure units
106	psf	pressure units
201	(absolute) pressure ratio	TABULAR, TURBINE, COMPRESS, or COMPPD only
202	Martin Beta Factor	TABULAR only

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 8. These units will apply to G values. See also MREF and NAF for further modifiers of these basic units.

TABLE 8. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
0	default: m^3/s if uid=si, ft^3/hr if uid=eng	
1	m^3/s	
2	m^3/min	
3	m^3/hr	
4	L/s	
5	L/min	
6	L/hr	
7	cm^3/s	

TABLE 8. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
8	cm^3/min	
9	cm^3/hr	
10	mm^3/s	
11	mm^3/min	
12	mm^3/hr	
13	kg/s	mass flow rate units
14	kg/min	mass flow rate units
15	kg/hr	mass flow rate units
101	ft^3/s (CFS)	
102	ft^3/min (CFM)	
103	ft^3/hr (CFH)	
104	gal/s (US GPS)	
105	gal/min (US GPM)	
106	gal/hr (US GPH)	
107	in^3/s	
108	in^3/min	
109	in^3/hr	
110	lb_m/s	mass flow rate units
111	lb_m/min	mass flow rate units
112	lb_m/hr	mass flow rate units

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then pressures and other state variables are defined on the basis of total or stagnation conditions, and either AF or both AFI and AFJ are required inputs. By default ISTP=TT, and so both inlet and outlet states are based on total (stagnation) conditions and therefore flow areas (AF or both AFI and AFJ) are required. ISTP=TS (total-static) and ISTP=ST (static-total) are also available as options. ISTP also affects the meaning of TREF and PREF.

Specific Speed Conversion Factor (SSCF) - A unit conversion factor (defaulting to unity) that applies to the output variable CNS (current compressor specific speed). The current specific speed for the compressor is available as an output variable, CNS, defined as follows:

$$CNS = SSCF*SPD*GCOMP^{1/2}/\Delta h^{3/4}$$

In the above definition, SSCF is a user-provided unit conversion factor which defaults to 1.0, SPD is the current speed, GCOMP is the current flow rate in units consistent with GU, GCF, MREF, and NAF (in other words, in the same units as any table values), and Δh is the ideal minimum increase in enthalpy across the compressor (e.g., at constant entropy, as if EFFP were unity). The Δh term has units of either J/kg or BTU/lb_m depending on model units (UID).

In "lazy" or quasi-non-dimensional English units, where SPD is in rpm, GCOMP is in ft^3/sec (GU=101, NAF=0, MREF=0,2, or 3... but not 1, and GCF=1.0), then the following converts BTU to ft-lb_f for traditional units

```
SCCF = 778.**(-0.75)
```

Auto Convert Torque - Checkbox to invoke automatic torque conversion (applicable only if the Speed in RPM box is checked). User may uncheck this box to activate the TCF field and assume control of the torque conversion factor. If checked, the TCF field will gray out and be defined by Sinaps based on the model units (UID). If using SI units, the TCF field will contain the conversion of rpm to rad/sec, resulting in TORQ units of N-m:

```
TCF = 60./(2.*pi)
```

In US Customary units, torque will be converted into lb_fft using:

```
TCF = 778./(2.*pi*60.)
```

TORQ Correction (TCF) - Correction factor for the output variable TORQ. The code calculates the hydraulic torque as:

```
TORQ = TCF*QTMK/SPD
```

where TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity. TORQ is a translatable path output variable.

Library Option - The library option allows the creation of a user-defined component library. The library saves all information in the current form to a database which can then be accessed and used with any model (including those with different units). It is the user's responsibility to define library locations, library names, and to control access to the library. If a library item is updated, the changes will not automatically propagate to a model, rather the user must go into the library dialog and reapply the library item to bring the updates into each device in a model. If sharing a library, be aware that other users can add, redefine, and delete entries in the library. A common library can be used for all Sinaps components or individual libraries can be created, e.g. a pump library, a compressor library, a pipe library, etc.

Select Library - Use this button to browse to, or create, a library file (*.xml format). When creating a new library file, click "Browse" to open browser window, browse to a location for the new library file, and type a name into the "File Name" field, e.g. turbomachine_library or turbomachine_library.xml. This file can be saved locally or on a network server. Once a library has been selected, the path and name of the library will appear in the adjacent field.

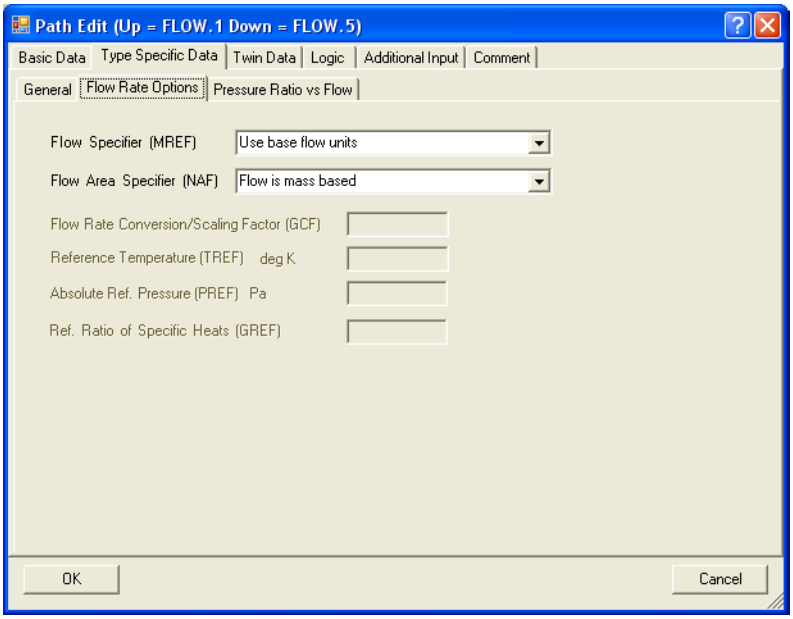
Library Selection - This field provides a pull-down list of all components defined in the selected library. If the library is empty, no entries will be listed.

Add to Library - Use the Add to Library button to add the current component description to the library. When prompted for a name, provide a unique name. The current configuration of the component will be saved. Subsequent changes to the component in the model will *not* change the saved library item. To update a component in the library, edit it in the model and then return to the library dialog and select the saved component name from the library and use the Add to Library button to save the updated component to the library under the same name. For a different name, simply provide a new name when prompted with the Component Name dialog and both prior and updated descriptions will be available in the library.

Use Selection - This button populates the component form with information that had been previously saved in the library. After a library has been specified and the Library Selection pull down has be used to select which component is to be imported, use the Use Selection button to import data to the current form. All fields will be imported.

Flow Rate Options Tab. The base units for the flow rate, G, are defined using the GU parameter whose meaning is summarized in Table 8 on page 84. However, the actual flow rate value provided in the tables, G_{table}, may deviate from the units specified by GU. The Flow Rate Options form (Figure 5-34 on page 87) provides access for specifying how flow rate is being defined if not based on the units specified by GU.

FIGURE 5-34. Compressor Flow Rate Options Form



Flow Specifier (MREF) - Integer mass flow reference flag. By default MREF is set to base flow units or zero, and hence G values provided are assumed to be based on the units specified by GU.

*Mass corrected (GCF*G*T**0.5/P)*, or MREF=1, instead means that the G values are actually corrected mass flows: the values supplied as "G" are actually to be interpreted as $G_{table} = GCF*G_{GU}*T^{1/2}/P$ where T and P are the absolute inlet temperature and pressure, respectively (static or total, depending on ISTP), and G_{GU} is determined by the unit selector GU. Absolute means the units of T are either K or R (depending on UID), and the units of P are either psia or absolute Pascal. GCF may be used for unit conversions or scaling.

*Mass corrected (GCF*G*(T/Tref)**0.5/(P/Pref))*, or MREF=2, $G_{table} = GCF*G_{GU}*(T/T_{ref})^{1/2}/(P/P_{ref})$ where the reference temperature and pressure (TREF, PREF) are specified separately.

Mass corrected function of Tref, Pref, Gref, or MREF=3, a more complicated correction is made for equivalent conditions (see main text) using a reference temperature and pressure and also a reference ratio of specific heats, γ_{ref} (GREF).

*Mass corrected (GCF*G*(RT/g)**0.5/P)*, MREF=4 invokes a slightly more complicated variation of MREF=1 that includes gas properties and unit conversions: $G_{table} = GCF*G_{GU}* (RT/g_c)^{1/2}/P$ where R is the gas constant, and g_c=1 if UID=SI or g_c=32.174 if UID=ENG. MREF may be used in combination with NAF.

Flow Area Specifier (NAF) - Integer mass flux basis flag. By default flow is mass based (NAF=0), and hence G values provided are assumed to be based on the units specified by GU.

Mass flux based on average area (NAF=1) instead means that the G values are actually mass fluxes based on the flow area AF, with the parameter GCF available for conversions or scaling. In other words, for NAF=1 the values supplied as "G" are actually to be interpreted as $G_{table} = GCF*G_{GU}/AF$ where G_{GU} is determined by the unit selector GU.

Mass flux based on upstream area (NAF=2), Gtable = GCF*GGU/AFI

Mass flux based on downstream area (NAF=3), Gtable = GCF*GGU/AFJ

(Note "G_{table}" is equivalent to "GCOMP" when using the GCOMP= method.)

Conversion factor applied to flow rate (GFC) - A G conversion factor (defaulting to unity) that is active if either NAF or MREF is nonzero. *GCF is ignored if MREF=NAF=0.*

Reference Temperature (TREF) - The reference temperature (K or R, depending on UID, and static or total, depending on ISTP) used for MREF=2 or MREF=3.

Absolute Reference Pressure (PREF) - The reference pressure (absolute Pascal or psia, depending on UID, and static or total, depending on ISTP) used for MREF=2 or MREF=3.

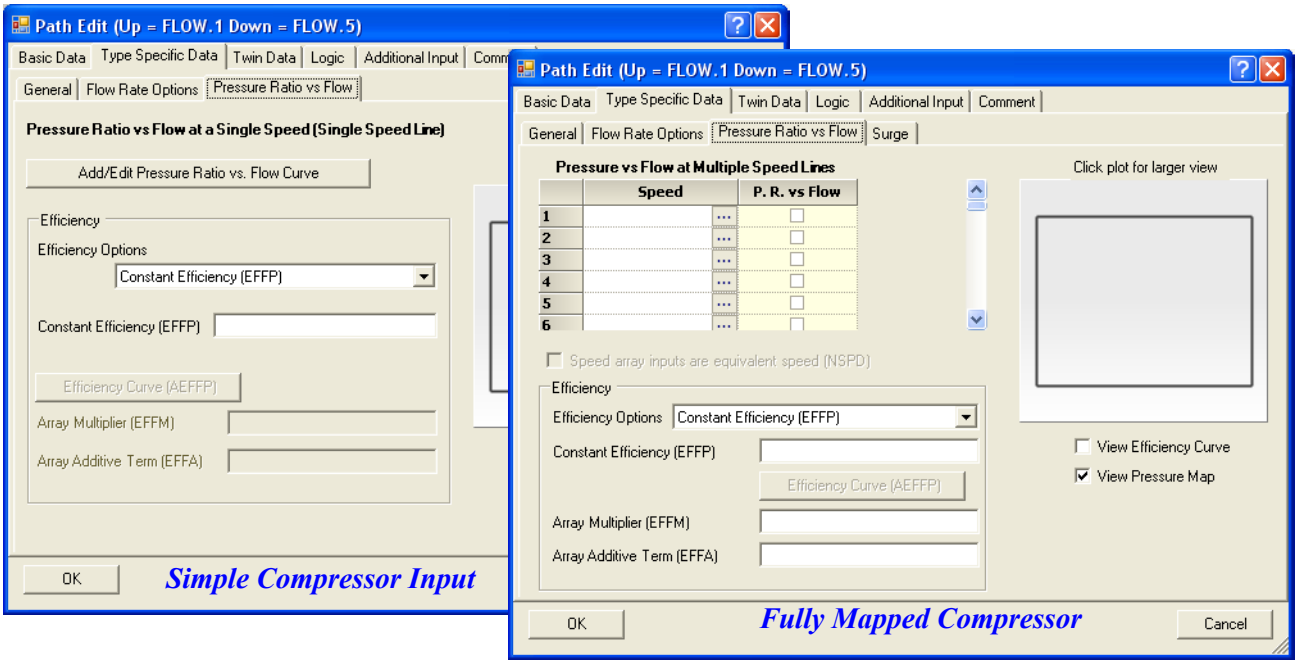
Reference Ratio of Specific Heats (GREF) - The reference ratio of specific heats, $\gamma_{ref} = Cp/Cv$, used for MREF=3.

Compressor Pressure vs Flow Tab. The contents of the Pressure vs Flow tab will vary depending on which type of compressor is being created, a simple or a fully mapped compressor. In both cases, Sinaps tables are used to input the performance and efficiency data. Once input, Sinaps will create the proper arrays to pass into SINDA/FLUINT when running the model. Note that tables created by compressors (and other turbomachinery devices) will not appear in the Sinaps Table Manager since they are only editable through the device dialog, in this case the COMPRESS edit form. Performance curves are unique for each device so although two paths may have identical performance curves, they cannot share a performance curve (updating one will not change the second).

Warning: Tables are used to input turbomachinary data. In general Tables explicitly recognize integer (treated as such in logic) and real numbers. For turbomachinary data inputs, data must be input as real numbers. i.e. you must include the decimal point. Inputting data as an integers will result in an error when executing a case.

Hint: If the model contains multiple compressors all based on the same performance curve(s), rather than creating each one individually, it is recommended to create one compressor, save it to the component library and then import the data from the library into the other compressor forms.

FIGURE 5-35. Compressor Pressure Ration vs Flow Tab



Simple, Single Curve Input - A simple compressor is a device which can be defined by a single pressure vs. flow curve.

Add/Edit Pressure Ratio vs Flow Curve (HG method) - The pressure ratios need not decrease monotonically, but the flow values must increase monotonically. The initial point is assumed to represent the surge or stall line, and the last point is assumed to represent the choke line.

Fully Mapped Compressor Inputs - A full map alternative to the single-curve, this is referred to as the HLIST method. The performance is defined by a series of pressure vs flow curves for a family of speeds. The initial points at each speed are assumed to represent the surge or stall line, and the last points at each speed are assumed to represent the choke line. However, the surge line may be specified independently using the GSURGE or HSURGE methods.

Pressure vs Flow at Multiple Speed Lines - The “Speed” table allows the pressure vs flow map to be defined. Input the individual speeds in the left column (monotonically increasing as you descend in the table). For each speed input, a pressure ratio versus flow table must be defined. The performance table for *all speeds* must be the same length. The pressure ratios need not decrease monotonically, but the flow values in each table must increase monotonically. The speeds in the speed are used by Sinaps to create the SPEEDS array.

After inputting a speed value click the adjacent [...] button to access the table input dialog. Note that the format of the Speed table will change depending on various options selected in the form. For example, selecting a full efficiency map (ELIST) in the Efficiency Options will add an efficiency column to the right of the P.R. vs Flow column. When the pressure ratio versus flow table input has been completed for each speed the check box in the P.R. vs Flow column will automatically toggle to a checked state indicating data has been defined.

When creating the performance table input at each speed, note the submodel, table ID, and structure are automatically assigned. A name is automatically assigned to the table, however the user can arbitrarily change this

value. Table conversion/scaling options can be used but the resolved values must be in units consistent with COEFS, HU and GU. From these various table inputs, Sinaps will automatically create the HLIST, GLIST, and individual H and G curves in the proper format to pass into SINDA/FLUINT. If a full efficiency map is defined, the ELIST and associated efficiency curve arrays will also be created.

Speed array inputs are equivalent speed (NSPD) - A flag to indicate the speeds in the table have been defined as equivalent speeds (scalable) instead of absolute speed. This option is only applicable to fully mapped pumps and dependent on the value of the *Flow Specifier (MREF)* on the Flow Rate Options tab (MREF = 3). If unchecked, NSPD defaults to 0 and speeds in the table are actual (e.g., rpm), if checked NSPD is set to 1 the speeds in the table are considered to be equivalent speeds. If absorbed power is supplied instead of isentropic efficiency (PLIST option) then the power in the PLIST tables is assumed to be scalable as an equivalent power as well. For more information see section 3.5.16.2 in the SINDA/FLUINT User's Manual.

Efficiency Options - There are multiple options for defining efficiency, a constant value of efficiency (EFFP), a single curve of efficiency versus flow (AEFFP), or a full efficiency map (only applicable for the full map option). Efficiencies are optional but should be specified or at least estimated. Efficiencies are used to calculate heating and torque. Alternatively in lieu of efficiency, the added or absorbed power can be mapped.

Constant Efficiency (EFFP) - Efficiency as a single user defined value (typical values are between 0.6 and 0.8, for example).

Efficiency Curve (AEFFP) - This button brings up the table dialog in which the submodel, number and structure are automatically defined by Sinaps. The user must input two columns of data defining efficiency (between 0 and 1) and flow. End values will be used if the provided range is exceeded; no extrapolations are made for this variable. Table conversion/scaling options can be used but the resolved values must be in units consistent with GU. Sinaps will use the data in the Efficiency Curve table to create the AEFFP array input for SINDA/FLUINT.

Full Efficiency Map - When a full efficiency map is selected, an efficiency column will be added to the Speed table. To input the efficiency data, click the [...] button next to each speed cell and edit the table to add the isentropic efficiency values (between 0 and 1). Once the data has been defined for each speed, the check box in the Efficiency column of the Speed table will toggle to a checked state indicating content. Sinaps will use the efficiency map data to create the ELIST array and the associated efficiency arrays at each speed.

Map of Absorbed Power (PLIST) - If a full performance map has been defined, instead of efficiencies, a map of added or absorbed power (W or BTU/hr, depending on UID) can be provided. In this case efficiency (EFFP) becomes an output variable. To use the PLIST option, enable the Map of Absorbed Power check box. To input the power data, click the [...] button next to each speed cell and edit the table to add the added or absorbed power values. Once the data has been defined for each speed, the check box in the Power column of the Speed table will toggle to a checked state indicating content. Sinaps will use this data to create the PLIST array and the associated power arrays for each speed.

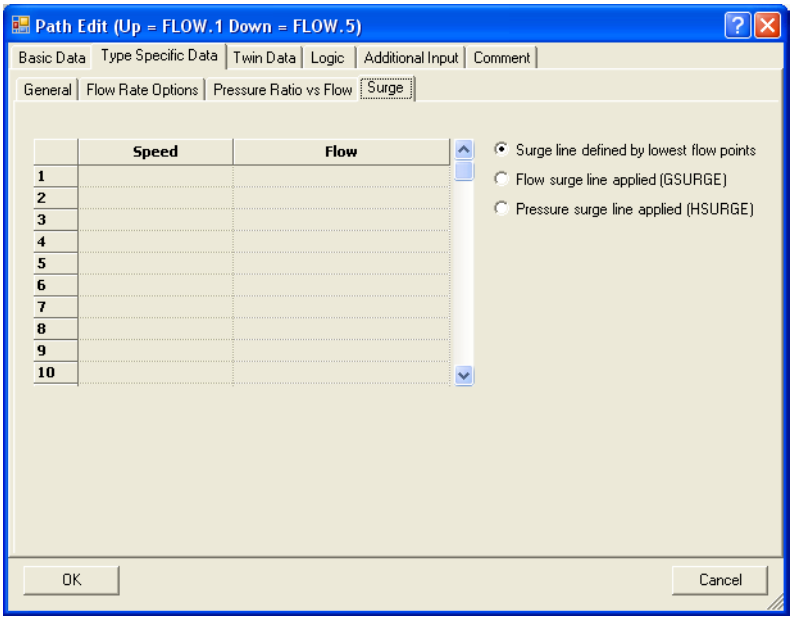
Array Multiplier (EFFM) - Multiplier to use for EFFP values extracted from array options (ELIST, AEFFP, or PLIST).

Array Additive Term (EFFA) - Additive term to use for EFFP values extracted from array options (ELIST, AEFFP, or PLIST).

Data Plots - Whether defining a simple or fully mapped device, the plot area will update automatically to show either a single performance curve or a family of curves (i.e. performance map) along with efficiency curves. Check boxes are provided to visibility toggles for the pressure vs flow plot and for the efficiency plot. Clicking on a plot will open the plot in the EZXY Plotter for a larger view.

Compressor Surge Tab. The surge tab shown in Figure 5-36 is used to define the surge line of a compressor. By default, the lowest flow points at each speed line are assumed to represent the surge line. Alternatively the user can independently specify a surge line, either as a function of flow (GSURGE) or pressure ratio (HSURGE), with one surge point for every speed in the SPEEDS array. If either the HSURGE or GSURGE options are selected, the table will activate and the speed column will automatically populate with the speed data from the Pressure vs Flow tab. The user must then specify only the pressure ratio or the flow, which ever is applicable. For more information on defining surge lines, see section 3.5.16.8 of the SINDA/FLUINT User's Manual.

FIGURE 5-36. Compressor Surge Options Form



5.4.5.12 Positive Displacement Compressor Device (COMPPD)

A COMPPD connector is a component-level model of a piston, rotary or reciprocating vane, screw, or scroll compressor. These machines differ from variable-displacement compressors (see COMPRESS in Section 5.4.5.11) in that a fixed volume of working fluid is, in principle, delivered with each revolution. While a COMPPD connector is normally used to represent an entire compressor containing any number of stages, it may also be used to model a single compressor stage. Such subset models may be combined in series for multi-stage compressors.

In order to simulate such a compressor or compressor stage, the user must specify a volumetric efficiency, η_v, as a function of pressure difference or ratio (H) and perhaps rotational speed (SPD). The default H is unitless: a pressure ratio (whether total to total, or total to static). Using the parameter name EFFV for η_v, and using DISP as the displacement volume per revolution, the volumetric flow rate is then calculated as:

G = GCF*EFFV*DISP*SPD

with GCF being a unit conversion factor that is specified by the user (and which defaults to unity).

A single curve of volumetric efficiency (at a single speed) may be specified as a function of pressure ratio, or a family of such curves can be defined as a function of speed. A single (constant) value of EFFV may also be specified, perhaps as adjusted in logic or expressions.

The base units for the flow rate, G, are defined using the GU parameter whose meaning is summarized in Table 4. Normally, G should have units of volumetric flow rate for COMPPD devices.

The program calculates G (available in logic or expressions as "GCOMP") as:

```
G = GCF*EFFV*DISP*SPD
```

The user supplies, directly or indirectly, all of the values on the right hand side of the above equation. EFFV is the volumetric efficiency (normally unitless), DISP is the volumetric displacement per revolution (normally in units of volume), and SPD is the shaft or rotor speed (normally in RPM). GCF is a unit conversion factor also supplied by the user such that the units of G match those specified via GU.

For example, if GU=1 (m^3/sec), if DISP is in units of cc, if EFFV is unitless, and if SPD is in RPM, then GCF = 1.0e-6/60. If instead DISP were supplied in units of cubic meters, then GCF=1/60 to convert RPM to revolutions per second.

If GU had specified mass flow units instead of volumetric flow units, then GCF would contain the inlet density (perhaps using "DL#up").

In summary, the units of EFFV, DISP, and SPD are unknown to the code, and can be any consistent set as long as GCF provides the conversions into the units specified by GU. (SPD should be consistent with TCF, an analogous factor used in torque calculations).

The "power efficiency" (*isentropic* efficiency, EFFP) may be defined as a constant, or as a function of pressure ratio, or as a function of both speed and pressure ratio. As an alternative, the power added to the flow can be specified.

The choices for specifying the pressure-flow relationships, via volumetric efficiency, include:

1. A single curve of volumetric efficiency versus speed. This method will be referred to as "the AEFFV= method."
2. A set of curves, each at increasing rotational speed. Within each speed curve, heads (pressure ratios) must increase monotonically. This method will be referred to as "the ELISTV= method."
3. If the pressure-flow maps exist as functions or are provided via calls to (perhaps COM-linked external) compressor design codes, then the current value of volumetric efficiency can provided directly as the parameter EFFV. In other words, the user can omit performance arrays and can instead manipulate the compressor parameters directly in logic and/or expressions. The variable for H is HCOMP. The code provides the current value of HCOMP (head, pressure drop, or pressure ratio), and the user must supply the corresponding values for EFFV. When using this method, the inlet fluid should not be an incompressible liquid, otherwise this COMPPD device will exhibit MFRSET-like insensitivity to pressure changes during the next solution interval. This method will be referred to as "the EFFV= method."

Compressor isentropic efficiencies are optional but should be specified or at least estimated (typical values are between 0.3 and 0.6, for example). Efficiencies are used to calculate power added and torque, and may be specified as either total-total or total-static (matching the pressure method described above via the ISTP parameter).

There are four ways to specify isentropic (or "power") efficiency, η_p, or alternatively, the extracted power:

1. as a single value, EFFP (updated in logic or expressions). This method will be referred to as "the EFFP= method."
2. as an interpolated array of isentropic efficiency vs. speed. This method will be referred to as "the AEFFP= method."
3. if a full map of *volumetric* efficiency EFFV has been provided via the ELISTV= method, then a full map of *isentropic* efficiencies can be provided in parallel (e.g, full maps of η_p at the same operating points as the pressure-η_v relationships). This method will be referred to as "the ELIST= method."
4. if a full map of *volumetric* efficiency EFFV has been provided via the ELISTV= method, then a full map of added or absorbed power (W or BTU/hr, depending on UID) can be provided in parallel (e.g, full maps of power at the same operating points as the pressure-η_v relationships). In this case, EFFP becomes an output variable. This method will be referred to as "the PLIST= method." Outside of the supplied range, scaling laws are applied to values of power (PLIST), unlike other variables which are interpolated linearly

The code calculates the hydraulic torque as:

```
TORQ      = TCF*QTMK/SPD
```

where TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity.

TORQ is a translatable path output variable. In SI units, the TCF contains the conversion of rpm (assuming those are the units of SPD) to rad/sec, resulting in TORQ units of N-m:

```
TCF = 60./(2.*pi)             $ "pi" is a built-in constant
```

In US Customary units, QTMK is in BTU/hr, so assuming SPD is in RPM, to get TORQ in lb_f-ft use:

```
TCF = 778./(2.*pi*60.)        $ 778 converts BTU/hr to lbf-ft/hr
```

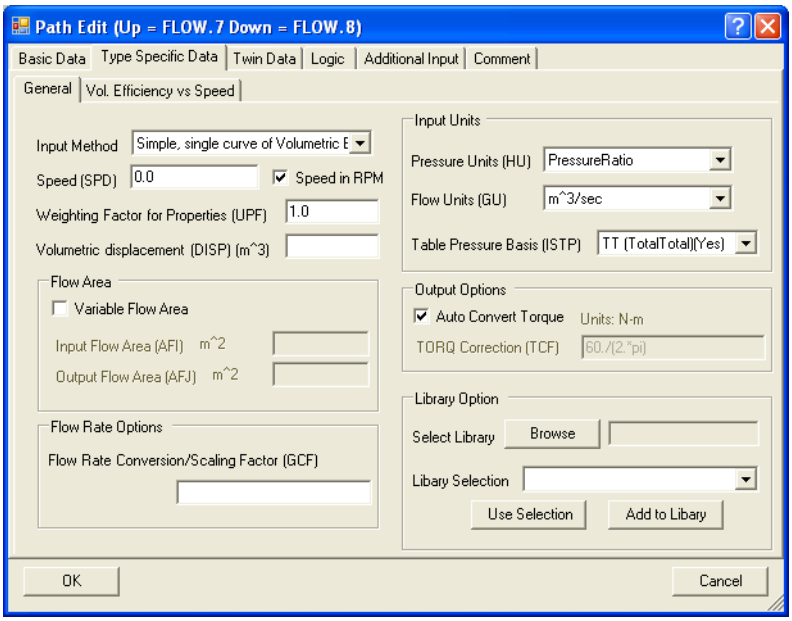
Hydraulic torque may be used along with co-solved ordinary differential equations (e.g., DIFFEQ1) to calculate the current speed of the compressor during transients, or may be used to solve a torque balance equation for steady-states (e.g., for turbobrayton cycles or turbochargers). Note that TORQ is positive for energy into the fluid system. It is therefore usually positive for compressors, and the negative of TORQ (usually a positive number) should be set proportional to $d\omega/dt$, the rate of change of shaft speed.

Figure 5-37 shows the type-specific edit form for a COMPPD device. Descriptions of each input are described below. For detailed information on describing compressors, torque, heating, and other advanced options, please refer to section 3.5.17 of the SINDA/FLUINT User's Manual.

> Note: The Basic Data tab has an option to "Use 5.3 Type Specific Data". The COMPPD form has been redesigned in version 5.4 to provide increased "user friendliness", assisting the user in determining which inputs are required and providing access to Sinaps Tables for array data input. When a 5.3 or older model is opened, the "Use 5.3 Type Specific Data" option will be checked and the Type Specific Data tab content will appear has the heritage style input form. The user may continue using the heritage form however it is recommended to update to the new form by unchecking the box. Be aware that when updating to the new form, some data (such as performance curves) will need to be reinput. For information on the heritage style form please see "Heritage Positive Displacement Compressor Form" on page A-11.

COMPPD General Tab. Data input on the General Tab applies to all positive displacement compressors regardless of the input method. For additional information, see "Positive Displacement Compressor Device (COMPPD)" on page 5-91.

FIGURE 5-37. COMPPD Type Specific General Form



Input Method - A COMPPD device can be defined as either a "Simple, single curve of Volumetric Efficiency vs. Speed (AEFFV method)" using a single curve to defined volumetric efficiency as a function of rotational speed, or a "Fully Mapped of Volumetric Efficiency vs. Pressure Ratio (ELISTV method)" which uses a family of curves to define the volumetric efficiency as a function of the pressure ratio at various speeds. The input method selection controls the content of the Vol. Efficiency vs Pressure Ratio tab allowing only applicable data fields to be displayed.

Current/Initial Speed (SPD) - Current/initial compressor speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency. Speed is used for TORQ calculations.

Speed in RPM - This check box allows automatic conversion of output torque. If the current/initial speed is not in units of RPM, uncheck this box. By default this box is checked since the speed units are typically input as RPM. When checked, the Auto Convert Torque box is activated, invoking automatic conversions to be applied to the output torque based on the model units (UID).

Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density.

Volumetric displacement (DISP) - Volumetric displacement of the compressor, usually in units of volume (per rotation), but any units can be used as long as the resulting G is consistent with the inputs GCF, GU, SPD, and EFFV (which is normally unitless).

Variable Flow Area - By default the flow area is defined as a constant area as defined by the field Flow Area (AF) on the General tab of the form. Checking the Variable Flow Area box enables a variable flow area across the path. When checked both the inlet (AFI) and outlet (AFJ) flow areas must be provided by the user and AF will be calculated automatically by the code.

Inlet flow area (AFI) - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Flow Rate Conversion/Scaling Factor (GFC) - A G conversion or scaling factor (defaulting to unity) which is applied to the calculated flow rate: GCOMP = GCF*EFFV*DISP*SPD.

Head unit identifier (HU) - Integer flag for head units: see Table 9. These units will apply to H values.

TABLE 9. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
0	default: m if uid=si, ft if uid=eng	
1	m	
2	cm	
3	mm	
4	cmH_2O, 4°C	pressure units
5	mmH_2O, 4°C	pressure units
6	mmHg, 0°C (Torr)	pressure units
7	Pa	pressure units
8	kPa	pressure units
9	Bar	pressure units
101	ft	

TABLE 9. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
102	in	
103	inH_20 (39.2°F)	pressure units
104	inHg (32°F)	pressure units
105	psi	pressure units
106	psf	pressure units
201	(absolute) pressure ratio	TABULAR, TURBINE, COMPRESS, or COMPPD only
202	Martin Beta Factor	TABULAR only

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 10. These units will apply to G values. See also MREF and NAF for further modifiers of these basic units.

TABLE 10. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
0	default: m^3/s if uid=si, ft^3/hr if uid=eng	
1	m^3/s	
2	m^3/min	
3	m^3/hr	
4	L/s	
5	L/min	
6	L/hr	
7	cm^3/s	
8	cm^3/min	
9	cm^3/hr	
10	mm^3/s	
11	mm^3/min	
12	mm^3/hr	
13	kg/s	mass flow rate units
14	kg/min	mass flow rate units
15	kg/hr	mass flow rate units
101	ft^3/s (CFS)	
102	ft^3/min (CFM)	
103	ft^3/hr (CFH)	
104	gal/s (US GPS)	
105	gal/min (US GPM)	
106	gal/hr (US GPH)	

TABLE 10. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
107	in^3/s	
108	in^3/min	
109	in^3/hr	
110	lb_m/s	mass flow rate units
111	lb_m/min	mass flow rate units
112	lb_m/hr	mass flow rate units

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then pressures and other state variables are defined on the basis of total or stagnation conditions, and either AF or both AFI and AFJ are required inputs. By default ISTP=TT, and so both inlet and outlet states are based on total (stagnation) conditions and therefore flow areas (AF or both AFI and AFJ) are required. ISTP=TS (total-static) and ISTP=ST (static-total) are also available as options.

Auto Convert Torque - Checkbox to invoke automatic output torque conversion (applicable only if the Speed in RPM box is checked). User may uncheck this box to activate the TCF field and assume control of the torque conversion factor. If checked, the TCF field will gray out and be defined by Sinaps based on the model units (UID). If using SI units, the TCF field will contain the conversion of rpm to rad/sec, resulting in TORQ units of N-m:

```
TCF = 60./(2.*pi)
```

In US Customary units, torque will be converted into $lb_f ft$ using:

```
TCF = 778./(2.*pi*60.)
```

TORQ Correction (TCF) - Correction factor for the output variable TORQ. The code calculates the output torque as:

```
TORQ = TCF*QTMK/SPD
```

where TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity. TORQ is a translatable path output variable.

Library Option - The library option allows the creation of a user-defined component library. The library saves all information in the current form to a database which can then be accessed and used with any model (including those with different units). It is the user's responsibility to define library locations, library names, and to control access to the library. If a library item is updated, the changes will not automatically propagate to a model, rather the user must go into the library dialog and reapply the library item to bring the updates into each device in a model. If sharing a library, be aware that other users can add, redefine, and delete entries in the library. A common library can be used for all Sinaps components or individual libraries can be created, e.g. a pump library, a compressor library, a pipe library, etc.

Select Library - Use this button to browse to, or create, a library file (*.xml format). When creating a new library file, click "Browse" to open browser window, browse to a location for the new library file, and type a name into the "File Name" field, e.g. turbomachine_library or turbomachine_library.xml. This file can be

saved locally or on a network server. Once a library has been selected, the path and name of the library will appear in the adjacent field.

Library Selection - This field provides a pull-down list of all components defined in the selected library. If the library is empty, no entries will be listed.

Add to Library - Use the Add to Library button to add the current component description to the library. When prompted for a name, provide a unique name. The current configuration of the component will be saved. Subsequent changes to the component in the model will *not* change the saved library item. To update a component in the library, edit it in the model and then return to the library dialog and select the saved component name from the library and use the Add to Library button to save the updated component to the library under the same name. For a different name, simply provide a new name when prompted with the Component Name dialog and both prior and updated descriptions will be available in the library.

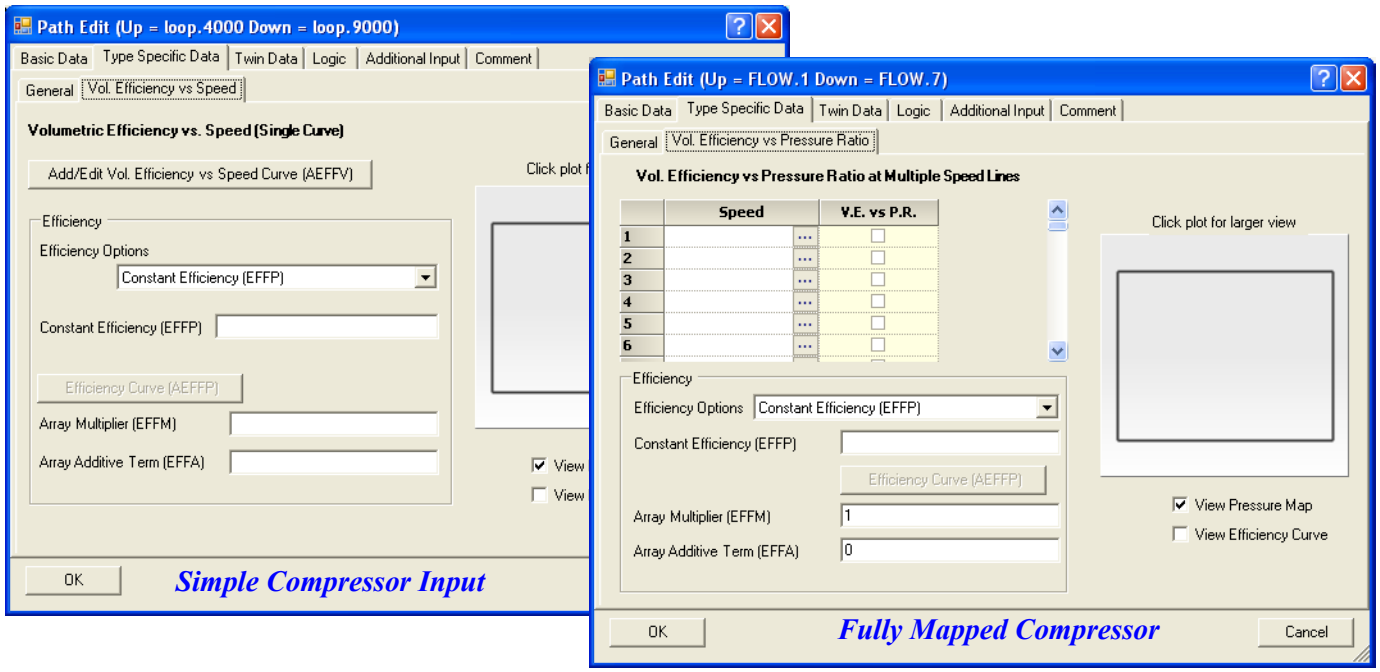
Use Selection - This button populates the component form with information that had been previously saved in the library. After a library has been specified and the Library Selection pull down has be used to select which component is to be imported, use the Use Selection button to import data to the current form. All fields will be imported.

COMPPD Volumetric Efficiency Tab. The contents of this tab will vary depending on which type of positive displacement compressor is being created, a simple (volumetric efficiency vs speed) or a full map (vol. efficiencies as a function of pressure ratio for multiple speed curves). In both cases, Sinaps tables are used to input the performance and efficiency data. Once input, Sinaps will create the proper arrays to pass into SINDA/FLUINT when running the model. Note that tables created by all turbomachinery devices will not appear in the Sinaps Table Manager since they are only editable through the device dialog, in this case the COMPPD edit form. Performance curves are unique for each device so although two paths may have identical performance curves, they cannot share a performance curve (updating one will not change the second).

Warning: Tables are used to input turbomachinary data. In general Tables explicitly recognize integer (treated as such in logic) and real numbers. For turbomachinary data inputs, data must be input as real numbers. i.e. you must include the decimal point. Inputting data as an integers will result in an error when executing a case.

Hint: If the model contains multiple compressors all based on the same performance curve(s), rather than creating each one individually, it is recommended to create one compressor, save it to the component library and then import the data from the library into the other compressor forms.

FIGURE 5-38. COMPPD Pressure Ratio vs Flow Tab



Simple, Single Curve Input - A simple positive displacement compressor is a device which can be defined by a single curve of volumetric efficiency vs speed.

> *Add/Edit Volumetric Efficiency vs Speed Curve (AEFFV method)* - Input a table of *volumetric* efficiency as a function of the actual (or equivalent) speed where speeds are monotonically increasing (i.e., $N_2 > N_1$). End values will be used if the provided range is exceeded; no extrapolations are made for this variable.

Fully Mapped Compressor Inputs - A full map alternative to the single-curve, this is referred to as the ELISTV method. The performance is defined by a series of volumetric efficiency versus pressure ratio curves for a family of speeds.

> *Volumetric Efficiency versus Pressure Ratio* - The "Speed" table allows the efficient vs. pressure ratio map to be defined. Input the individual speeds in the left column (monotonically increasing as you descend in the table). For each speed input, a pressure ratio versus efficiency table must be defined. The performance table for *all speeds* must be the same length. The efficiencies need not decrease monotonically, but the pressure ratio values in each table must increase monotonically. After inputting a speed value click the adjacent [...] button to access the table input dialog. When the table input has been completed for each speed, the check box in the VP.R. vs Flow column will automatically toggle to a checked state indicating data has been defined. The speeds in the speed are used by Sinaps to create the SPEEDS array.
>
> When creating the performance table input at each speed, note the submodel, table ID, and structure are automatically assigned. A name is automatically assigned to the table, however the user can arbitrarily change this value. Table conversion/scaling options can be used but the resolved values must be in units consistent with COEFS, HU and GU. From these various table inputs, Sinaps will automatically create the HLIST, GLIST, and

individual H and G curves in the proper format to pass into SINDA/FLUINT. If a full efficiency map is defined, the ELIST and associated efficiency curve arrays will also be created.

Efficiency Options - There are multiple options for defining efficiency, a constant value of efficiency (EFFP), a single curve of efficiency versus flow (AEFFP), or a full efficiency map (only applicable for the full map option). Efficiencies are optional but should be specified or at least estimated. Efficiencies are used to calculate heating and torque. Alternatively in lieu of efficiency, the added or absorbed power can be mapped.

Constant Efficiency (EFFP) - Efficiency as a single user defined value (typical values are between 0.6 and 0.8, for example).

Efficiency Curve (AEFFP) - This button brings up the table dialog in which the submodel, number and structure are automatically defined by Sinaps. The user must input two columns of data defining efficiency (between 0 and 1) and flow. End values will be used if the provided range is exceeded; no extrapolations are made for this variable. Table conversion/scaling options can be used but the resolved values must be in units consistent with GU. Sinaps will use the data in the Efficiency Curve table to create the AEFFP array input for SINDA/FLUINT.

Full Efficiency Map - When a full efficiency map is selected, an efficiency column will be added to the Speed table. To input the efficiency data, click the [...] button next to each speed cell and edit the table to add the isentropic efficiency values (between 0 and 1). Once the data has been defined for each speed, the check box in the Efficiency column of the Speed table will toggle to a checked state indicating content. Sinaps will use the efficiency map data to create the ELIST array and the associated efficiency arrays at each speed.

Map of Absorbed Power (PLIST) - If a full performance map has been defined, instead of efficiencies, a map of added or absorbed power (W or BTU/hr, depending on UID) can be provided. In this case efficiency (EFFP) becomes an output variable. To use the PLIST option, enable the Map of Absorbed Power check box. To input the power data, click the [...] button next to each speed cell and edit the table to add the added or absorbed power values. Once the data has been defined for each speed, the check box in the Power column of the Speed table will toggle to a checked state indicating content. Sinaps will use this data to create the PLIST array and the associated power arrays for each speed.

Array Multiplier (EFFM) - Multiplier to use for EFFP values extracted from array options (ELIST, AEFFP, or PLIST).

Array Additive Term (EFFA) - Additive term to use for EFFP values extracted from array options (ELIST, AEFFP, or PLIST).

Data Plots - Whether defining a simple or fully mapped device, the plot area will update automatically to show either a single performance curve or a family of curves (i.e. performance map) along with efficiency curves. Check boxes are provided to visibility toggles for the pressure vs flow plot and for the efficiency plot. Clicking on a plot will open the plot in the EZXY Plotter for a larger view.

5.4.5.13 Null Connector (NULL)

The NULL (no device) connector provides direct access to the underlying parameters of any connector, and is considered an extremely advanced and rarely needed option. Refer to Section 3.5.1 of the SINDA/FLUINT User's Manual for more details.

5.4.6 Heat Transfer Ties

Ties are special conductor-like elements used to link objects in fluid submodels with objects in thermal submodels. Specifically, a tie *always extends between a fluid lump and a thermal node* to permit convective energy interchange. In general, the lump represents the fluid within a container or pipe segment, and the node represents the inner wall of the container. Any number of ties may be attached to a lump or to a node. Tie parameters can be input as numeric values, registers, complex expressions, or conditional statements. They may also be defined based on processor variables.

Ties are unique because they are the only network element that can connect fluid submodels to thermal submodels, and they can only be used for that purpose. Although they are very similar to inter-submodel thermal conductors, ties are strictly fluid network elements and always "belong" to the fluid submodel in which they were input. In order to tie lumps in two different fluid submodels together (to model perhaps a thin-walled heat exchanger), there must exist an intermediate thermal node that represents perhaps the separating wall. Also, note that a constant wall temperature condition can be simulated by tying a lump to a boundary node; a constant heat flux condition is modeled with the QL term of the lump, and no ties are needed.

> Caution: When creating ties between a fluid lump and a thermal node, care must be taken during the creation of the adjacent thermal network. If a thermal node is tied to a fluid lump and connected *very strongly* to another thermal node, a solution for the tied node can become elusive. This will often result in smaller time steps and longer execution time, the inability of a model to converge, or in some unique cases convergence to an erroneous solution. When attaching to tied nodes in the thermal network, be sure the connection between the fluid lump and thermal node is significant when compared to the node's connection to other thermal nodes, perhaps setting the thermal conductance to no more than 100x the tie conductance. As always, use engineering judgment and external calculations to evaluate a solution and decide if it is reasonable or not.

For complete documentation and ties, types, usage, referencing in logic, etc. please see section 3.6 of the SINDA/FLUIN User's Manual.

Types of Ties: Ties are subdivided into types: *HTN, HTNC, HTNS, HTU, HTUS and HTP.* The first three types (HTNx) represent heat transfer options based on built-in forced convection correlations. The fourth and fifth types (HTUx) represent user-defined heat transfer processes. The HTP tie is used to model pool boiling or heat transfer to a nearly stagnant tank or vessel.

```
HTN . . . . . Convection heat transfer
HTNC. . . . . Convection heat transfer, centered lump
HTNS. . . . . Convection heat transfer, segment-oriented version of HTN
HTU . . . . . User-described heat transfer
HTUS. . . . . User-described heat transfer, segment-oriented version of HTU
HTP . . . . . Pool Boiling heat transfer
```

With HTN, HTNC, and HTNS ties, the conductance (UA value) is calculated automatically by the program according to convective heat transfer in a duct (see Appendix B and Section 7.10.3 of the SINDA/FLUINT User's Manual for details). The user must identify one or more adjacent paths (TUBEs or STUBE connectors) to provide the duct length, diameter, flow velocity, etc. to be used in the calculations. With HTU and HTUS ties, the user provides the UA coefficient either directly or via logic. When using the HTP tie option, based on the temperature profile the heat transfer will be automatically calculated as natural convection, pool boiling, or transition and film boiling.

Refer to section 3.6.2 of the SINDA/FLUINT User's Manual for in depth clarification on the various types of ties.

Centered Lump Ties: The user may "lump" all the associated mass and volume of a heat exchanger section into a tank (or other lump), and connect this tank to the up- or downstream lump with a single path that has the appropriate geometry of the section. This is the basis of the HTN tie. With only one path used when defining the tie, energy is assumed to enter or leave the section at one end or the other (i.e., either upstream or downstream). Flow reversals *do not change* the location at which heat is exchanged.

Alternatively, the user may "center" the lump between two paths to represent the heat exchanger section. This is the basis of the HTNC ("centered") tie. In other words, the energy entering or leaving the lump is the sum of the heat transfer calculated with *two paths* representing the container wall. One path may be upstream of the lump, and one may be downstream. For a string of such sections, the paths between the lumps may be shared (divided in two) for heat transfer purposes. By default the tie uses the average of the upstream and downstream paths. The user can change this weighting by adjusting the path fractions for wetted area.

Lump Oriented vs. Path-Oriented Ties: HTN, HTNC, and HTU ties calculate heat transfer assuming that the lump represents the bulk fluid properties, and that the node represents the average wall temperature. This lumped parameter formulation implicitly assumes downstream-weighted heat transfer, since the lump state changes according to the heat transfer rate, and only that state is used in calculating heat transfer. Adjacent paths are used only to provide flow rate and heat transfer area data; whether they are upstream or downstream of the lump is irrelevant. While these methods are robust and are recommended for flows with phase change, they tend to underpredict heat transfer rates slightly for flows without phase change, especially when the resolution is too coarse. Also, they tend to conflict with the way many engineers treat a single-phase heat transfer problem: as a constant wall temperature over a segment that has distinct inlet and outlet states.

HTNS and HTUS ties, which will be referred to as segment ties, treat the heat transfer problem differently than do lumped parameter ties. Instead of being lump-oriented, they are path-oriented. The node is assumed to represent the average wall temperature over the length of the path, the lumps on either end of that path are assumed to represent the upstream and downstream states of that segment. For single-phase flows, the segment tie is equivalent to an LMTD (log-mean temperature difference) approach. Segment ties should be associated with the path upstream from the lump to which the tie connects in the network. Unless the user specifies otherwise, these ties will automatically jump to the lump that is currently downstream of the specified path when flows reverse. The network diagram itself will not reflect any jumping of the connection for flow reversals in a segmented tie.

> Note: While HTNS ties are preferred in the limit of steady single-phase or flows without phase change, HTNC ties are recommended for unsteady flows with phase change or where flow conditions are not known a priori.

Customizing heat transfer calculations. For calculated heat transfer ties (HTN, HTNC, and HTNS) the user may extensively customize the heat transfer calculations for both laminar and turbulent flow as needed to model non-circular ducts, entrance length (developing thermal flow) effects, heat transfer enhancements such as internal fins, coiled tubes, enhanced heat transfer, cryogenic and near-critical fluids with strong property variations, etc.

For single-phase turbulent flow, the user can address all of the coefficients of the Dittus-Boelter coefficient, plus apply correction exponents, plus apply a multiplier (XNTM) and a bias (XNTA). Thus:

$$Nu_{turb} = XNTM * Nu_{DB} + XNTA$$

where for heating ($T_{wall} > TEF$):

$$Nu_{DB} = CDB * Re^{UER} * Pr^{UEH} * (m_r/m_{wall})^{UEV} * (r_r/r_{wall})^{UED} * (k_r/k_{wall})^{UEK} * (Cp_{avg}/Cp_{wall})^{UECP} * (P/P_{crit})^{UEP} * (T_r/T_{wall})^{UET}$$

For cooling ($T_{wall} < TEF$), the Prandtl exponent becomes UEC instead of UEH.

TEF is the effective fluid temperature. It is the same as the lump static temperature, TL, in the limit of low-speed single-phase flows. See the SINDA/FLUINT User's Manual for more details, including advanced high speed flow and two-phase flow options.

The following parameters are available to the user to customize the above Dittus-Boelter correlation:

CDB. . . . . Dittus-Boelter scaling parameter
UER. . . . . Reynolds number exponent
UEC. . . . . Prandtl number exponent for cooling
UEH. . . . . Prandtl number exponent for heating

Other parameters, XMTM, UED, etc. are also available for customization along with other correlations such as the laminar Nusselt number, the effective fluid temperature, and two-phase options. Section 3.6.6 of the SINDA/FLUINT User's Manual discusses these options in detail. The Additional Input tab can be used for defining the various input parameters.

User-defined heat transfer ties - For user-defined ties (HTU and HTUS), the user must provide an overall conductance for the tie. The user can define the conductance as a constant UA, as a register-based expression.

One alternate to the "constant UA" problem is to describe how the UA varies as a function of temperature difference, in which case the Temperature Dependent check box must be selected. This will activate the lower data fields in the form where the user must define the AHT, UB, and UEDT parameters as follows:

$$UA_{tie} = AHT_{tie} \cdot UB_{tie} \cdot (T_{node} - TEF_{tie})^{UEDT_{tie}}$$

where:

AHT. . . . . heat transfer area (m^2 or ft^2)
UB. . . . . the base (temperature-independent) portion of the conductance U
UEDT. . . . . exponent on temperature difference

Note that if UEDT=0.0, then the above "temperature-dependent" method degenerates into simply UA=UB*AHT, which is convenient to use because the UB can then be centrally defined (using a register or expression common to many ties), while each tie might apply a local value of the heat transfer area AHT.

Note also that the heat rate (QTIE) will be in proportion to $(T_{node} - TEF_{tie})^{1+UEDT_{tie}}$

Twinning Ties: When used with twinned tanks to model phasic nonequilibrium, ties themselves can also be twinned. The secondary twin is ignored unless the lump itself is twinned and currently active. If ties attach to twinned tanks and use twinned paths to calculate convection, the tie itself *must* be twinned. When a twinned tie attaches to a twinned tank, the primary tie attaches to the primary (liquid) tank, and the secondary tie attaches to the secondary (vapor) tank.

5.4.6.1 To Create a Tie

When creating ties in the network diagram, there are two methods as described below. After creating ties, they must be edited such that the required input parameters are defined.

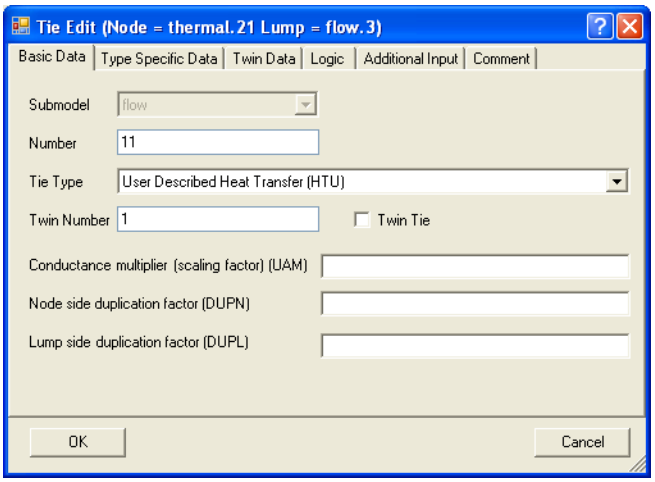
1. Fluid Icon Toolbar - Click on the heat transfer tie icon in the tool bar and then draw the link between a lump and a node and a tie will be created. Clicking on another tie type will change the type of tie to be created.
2. Tool Pallet - Select the type of heat transfer tie to be created and simply draw a link between a lump and a node and a tie will be created. Clicking on another tie type will change the tie type to be created. Clicking on the Tool Pallet background will unselect a tie type.

5.4.6.2 To Edit a Tie

A tie can be edited using either the Tie Edit form or through the Object Properties window. The Tie Edit form can be accessed by double-clicking on the tie or right-clicking on a tie to access the object edit menu and selecting Edit Tie. Alternatively, clicking on a tie will highlight it (grip points will become visible). Once selected, the tie data is displayed in the Object Properties window and can be edited directly. Note that when using the Object Properties dialog to edit a tie, input checking is bypassed.

Tie Basic Data Tab. The Basic Data tab shown in Figure 5-39 provides access to define the tie identification number, type of tie, twin options, a scaling factor, and duplication factors.

FIGURE 5-39. Tie Edit Form



Number - The identification number of the tie. Numbers are automatically assigned by Sinaps but can be overridden by the user. This value must be a positive integer value.

Tie Type - Select the type of tie to create, user defined (HTU, HTUS), calculated convection (HTN, HTNC, HTNS), or pool boiling (HTP).

Twin Option - To define a twinned tie, select the Twin Tie check box. The twin number will automatically be defined to avoid conflicts with existing numbering schemes. The user can override the automatic numbering by typing in a value. When a twinned tie attaches to a twinned tank, the primary tie attaches to the primary (liquid) tank, and the secondary tie attaches to the secondary (vapor) tank.

Conductance Multiplier or Scaling Factor (UAM) - Overall scaling factor (multiplier, defaults to unity). This can be used for sensitivity studies and parametrics by setting UAM to a register and then varying the register.

Duplication Factors (DUPN and DUPL) - Ties may be assigned duplication factors, which is a general device for exploiting symmetry to reduce the size of the model. The key to understanding duplication factors is this: *the lump "sees" a tie DUPL times, while the node "sees" the same tie DUPN times*; DUPL and DUPN need not be equal. Duplication factors do not need to be integers. Setting DUPL=DUPN=0.0 disables a tie, though the QTIE will not be zero.* For more information on duplication factors, see section 3.12 "Symmetry and Duplication Options" in the SINDA/FLUINT User's Manual.

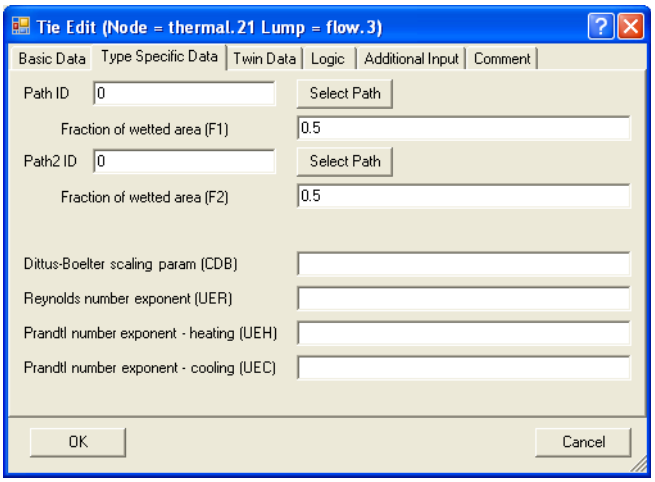
Type Specific Tie Data: The Type Specific Data tab varies based on the type of tie selected. This form contains fields to define the type specific parameters for a tie which will vary depending on which type is selected on the Basic Data tab. This tab only lists the most commonly used parameters for each type of tie. Less common input can be defined on the Additional Input tab.

Jump to type specific forms:

"Calculated Heat Transfer (HTN, HTNC, HTNS) Type Specific Data" on page 5-106
"User Described Heat Transfer (HTU, HTUS) Type Specific Data" on page 5-107
"Pool Boiling (HTP) Type Specific Data" on page 5-108

* By comparison, setting UAM=0.0 also disables a tie, and also causes QTIE=0.0

Calculated Heat Transfer (HTN, HTNC, HTNS) Type Specific Data.



Path1 ID (PID) - The path id of first path to use for flow rate and geometry (either TUBE or STUBE connector). The user *must* define an associated path, the centered option (HTNC) also requires a second path to be defined. To select the path, click the Select Path button, the form will disappear from the screen providing access to the network, and then select the path to be associated with the tie.

For the segment tie option (HTNS), the path selected must be *downstream* of the lump. The segment tie option has a check box option labeled Stay. If this option is checked, the tie (and its associated heat rate) will "stay put" on the downstream lump during the solution. If the box is not checked, the tie will move to the upstream lump if the flow through the path reverses. By default the Stay box is unchecked.

Fraction of Wetted Area (F1 and F2) - This is the fraction of pid wetted area or fouling factor attributed to tie. The user may change the fraction of wetted area for each path if desired, possibly to adjust for sharing of a path by multiple ties.

Path2 ID (P2ID) - The path id of second path to use for flow rate and geometry (either TUBE or STUBE connector). This option is only applicable only to the HTNC type of tie.

Customizing the Heat Transfer Correlation - For calculated heat transfer ties (HTN, HTNC, and HTNS) the user may extensively customize the heat transfer calculations for both laminar and turbulent flow as needed. The following parameters can be used to modify the Dittus-Boelter turbulent correlation shown in "Customizing heat transfer calculations" on page 5-102. For additional information please see sections 3.6.6 and 3.6.7 of the SINDA/FLUINT User's Manual.

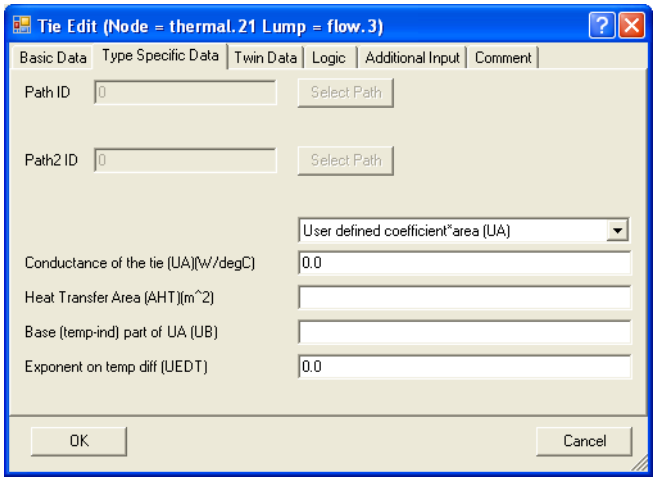
Dittus-Boelter Scaling Parameter (CDB) - Defaults to 0.023 if left blank.

Reynolds Number Exponent (UER) - Defaults to 0.8 if left blank.

Prandtl Number Exponent for Heating (UEH) - Defaults to 0.4 if left blank.

Prandtl Number Exponent for Cooling (UEC) - Defaults to 0.3 if left blank.

User Described Heat Transfer (HTU, HTUS) Type Specific Data.



Path1 ID (PID) - Applicable to segmented user ties (HTUS) only. The path id of the downstream path used to define the segment to which the tie is attached (can be any type of path). To select the path, click the Select Path button, the form will disappear from the screen providing access to the network, and then select the path to be associated with the tie. If the check box option labeled Stay is checked, the tie (and its associated heat rate) will "stay put" on the down-stream lump during the solution. If the box is not checked, the tie will moved to the upstream lump if the flow through the path reverses. By default the Stay box is unchecked.

HTU Type - To assist the user in defining HTU ties, Sinaps provide a pull down which activates the appropriate fields for the each sub-type of HTU. The sub-types are:

Type of HTU Tie	Inputs
User defined coefficient*area	Assumes a constant UA, the user must input the heat transfer coefficient times the area.
User defined coefficient and area	Assumes a constant UA, the user must input the heat transfer coefficient as UB and the area as AHT.
Temperature Dependent Coefficient	User must input AHT, UB and UEDT $UA_{tie} = AHT_{tie} \cdot UB_{tie} \cdot (T_{node} - TEF_{tie})^{UEDT}$
Logic Controlled	This option indicates the tie conductance is defined via logic, perhaps within the logic tab of the tie form. Input in the active data fields is optional.

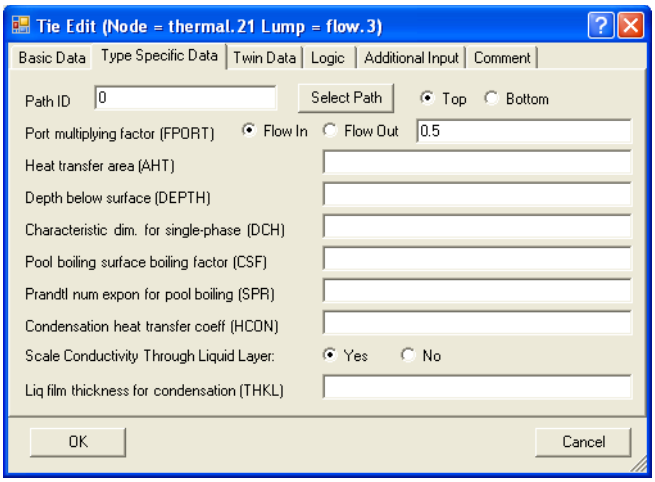
Conductance of the Tie (UA) - User defined conductance of the tie. The conductance must be positive or zero.

Heat Transfer Area (AHT) - User defined heat transfer area in units consistent with the model units (UID).

Base (temperature independent) Part of UA (UB) - The base (temperature-independent) portion of the conductance (U).

Exponent of Temperature Difference (UEDT) - User defined exponent on temperature difference across the tie.

Pool Boiling (HTP) Type Specific Data.



Path ID (IPORT) - The path ID (optional) of the vessel inlet or outlet flow path to use for flow rate and flow area (any path with a nonzero AF is valid) for single-phase calculations based on an internal call to NCCYLIN. If iport is missing or zero, natural convection within a closed sphere (NCSPHERE methods) will be used for estimating single-phase heat transfer. To select the path, click the Select Path button, the form will disappear from the screen providing access to the network, and then select the path to be associated with the tie.

Location Top/Bottom - The location of the port, either at the top or the bottom of the vessel. Only applicable if IPORT has been assigned. This radio button controls the sign of IPORT being passed into SINDA/FLUINT (positive for the top, and negative for the bottom of the vessel).

Port Multiplying Factor (FPORT) - Optional multiplier on the flow rate (not wetted surface area) of IPORT. For example, the associated path (IPORT) is assumed to flow into the vessel, setting FPORT= 1.0 and setting the Flow In/Flow Out radio button to be Flow Out, can be used to rectify the flow rate of the path if it instead flows out of the vessel in the positive FR direction.

Flow In/Flow Out - Fluid is assumed to be flowing into the vessel by default. This radio button controls the sign of FPORT being passed into SINDA/FLUINT.(flow in is positive, flow out is negative).

Heat Transfer Area (AHT) - User defined heat transfer area.

Depth Below Surface (DEPTH) - Depth of the wall node beneath the liquid/vapor surface (as represented by the lump state). Used in conjunction with ACCM as noted above to calculate the saturation temperature of the liquid against the wall. When ACCM is nonzero, DEPTH can also affect apportioning to twinned tanks. Negative values are valid, and are interpreted as meaning “this node is above the surface.” A value of zero means that this node is exposed to both phases.

Characteristic Dim. for Single-phase (DCH) - Characteristic dimension (in meters or feet) for natural convection and other single-phase (nonboiling, noncondensing) flows, typically the depth of below the fluid surface. If IPORT has not been input, the DCH is treated as the diameter of an effective sphere in an internal call to NCSPHERE. Otherwise, DCH is treated as the length of an effective cylinder in an internal call to NCCYLIN.

Pool Boiling Surface Boiling Factor (CSF) - Surface roughness factor C_{sf} for pool boiling. (Caution: values much lower than the default of 0.01 can result in very large heat transfer coefficients.)

Prandtl Num Exponent for Pool Boiling (SPR) - Prandtl number exponent S for pool boiling. Override the default with 1.0 when using water as the working fluid.

Condensation Heat Transfer Coefficient (HCON) - Condensing heat transfer coefficient (W/m^2-K or BTU/hr-ft^2-F).

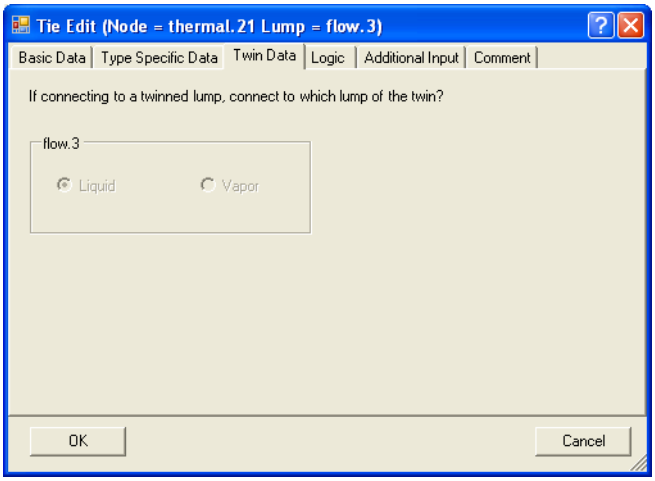
Scale Conductivity Through Liquid Layer - A flag to indicate HCON should be used as a scaling factor on the liquid conductivity based on $|HCON|*K_{liq}/THKL$ where K_{liq} is the thermal conductivity of the liquid. This flag controls the sign of HCON being passed into SINDA/FLUINT (yes is negative, no is positive).

Liquid Film Thickness for Condensation (THKL) - Thickness of liquid layer when condensing. Only applicable when HCON is used as a scaling factor.

Twin Data: When an untwinned (singular) tie is connected to a twinned tank, the user must specify to which side of the tank the tie should be attached. The twin data form shown in Figure 5-40 allows the user to select either the liquid or the vapor side of the tank. If it connects to the liquid (primary) side, the tie will remain attached to it at all times. If the tie connects to the vapor (secondary) twin, the tie will attach to it when the tank is split (in the nonequilibrium mode), but will otherwise move temporarily to the liquid twin when the tanks are in the combined (homogeneous) mode.

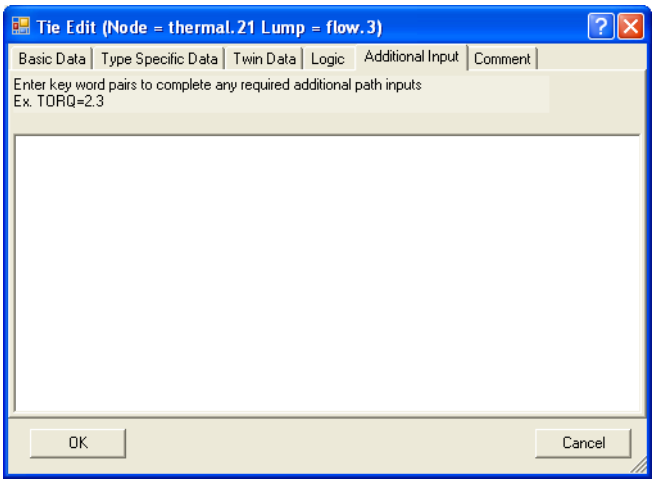
If the adjacent tank is not twinned the liquid/vapor options will be grayed out and a check box will be provided to twin the lump. Checking this box will twin the tank along with enabling the liquid/vapor dialog. If the adjacent lump is a junction or a plenum, the Twin Data dialog will be grayed out.

FIGURE 5-40. Tie Twin Data



Additional Input Tab: Many advanced tie parameters are not distinctly listed on the edit form since they are not commonly used and require advance usage knowledge. If the user wishes invoke any parameter discussed in Section 3.6 of the SINDA/FLUINT User's Manual which does not have an associated input field, the additional tab can be used. This form is a simply text box for inputting standard ASCII text format as shown in Figure 5-41. Since these inputs are column sensitive and should not start in column 1, Sinaps will automatically start the inputs in column 2. Therefore comments cannot be designated using a 'C' in column 1. However in-line '$' comments can be used.

FIGURE 5-41. Tie Additional Input Form



5.4.6.3 To Move a Tie

Tie can be relocated simply by clicking on the tie to highlight it and make the grip points visible. Next, drag the appropriate grip point from the existing lump or node a new lump or node.

5.4.6.4 Setting Tie Defaults

Tie defaults can be set using the Model Browser or the Model Defaults dialog. For more information, see “Setting Network Element Defaults” on page 5-6.

5.4.7 Fluid to Fluid Heat Transfer (FTIES)

Fties (fluid to fluid ties) may be used to establish heat transfer directly between lumps, and they are therefore directly analogous to linear SINDA conductors. However, fties cannot extend *between* fluid submodels: *they must attach to lumps within the same fluid submodel*. When they are used, fties are often defined parallel to tube or STUBE paths, and represent the axial conduction of the fluid within that duct segment. The paths model the fluid flow, while the ftie models heat transfer between lumps.

> Note: The automatic ftie generation option on paths (“Tube and STUBE Input” on page 5-44) is preferred over independent ftie creation for modeling axial conduction. The independent ftie option in this section is available for customization and for consistency.

There are two common uses for fties: (1) axial conduction or “back conduction” of heat through the fluid itself, which is otherwise ignored as negligible compared to advection, and (2) direct contact or thin-walled fluid-to-fluid heat exchange within the same fluid system, such as happens in regenerators. With the exception of highly conductive fluids (such as liquid metals and superfluid helium) or systems with extremely small flow rates (such as loop heat pipes), axial conduction is normally negligible: *fties are rarely required.*

5.4.7.1 To Create and FTIE

When creating fties in the network diagram, there are two methods as described below. After creating fties, they must be edited such that the required input parameters are defined.

1. Fluid Icon Toolbar - Click on the ftie icon ( ) in the tool bar and then draw the link between two lumps and the ftie will be created.
2. Tool Pallet - Select the ftie device on the Fluid>Other Tool Pallet, then simply draw a link between two lumps and the ftie will be created. Alternatively, drag the ftie icon onto the network drawing, and then click on the end grip points and move them to the appropriate lumps to connect.

When a ftie is created by dropping the icon from the Tool Pallet onto the diagram and then connecting it, the left side of the ftie is connected to lump C and the right side will be lump D. When created by drawing the link between two lumps, the “from” lump (where the link starts during creation) will be lump C while the “to” lump becomes lump D. These designations are used for defining positive heat flow through the ftie. Heat transfer will be positive when it leaves lump C and flows toward lump D. The C and D designation are also relevant to the ftie duplication factors (DUPC and DUPD).

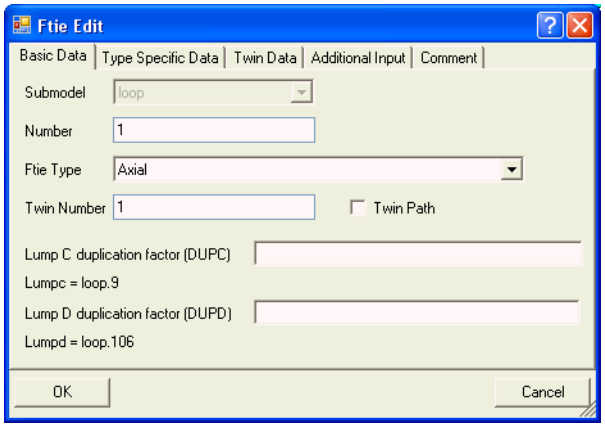
5.4.7.2 To Edit and FTIE

A ftie can be edited using either the Object Properties window or the Ftie Edit form. The edit form can be accessed by double-clicking on the ftie or right-clicking on a ftie to access the object edit menu and then selecting Edit Ftie. Alternatively, clicking on a ftie will highlight it (grip points will become visible). Once selected, the ftie data is displayed in

the Object Properties dialog and can be edited directly. For extensive information regarding fties, please see section 3.7 of the SINDA/FLUINT User's Manual.

The Ftie Edit form is a tabbed form providing access to various input options as shown in Figure 5-42. The Basic Data tab provides access to define the ftie identification number, type of ftie, and duplication factors.

FIGURE 5-42. Ftie Edit Form



Identification Number: The ID must be a positive numeric value.

Ftie Type: Three different types of Fties are available: User, Constant Q and Axial.

1. **User** fties are similar to HTU ties. The user has complete control over the conductance (GF, analogous to UA), and the resulting heat rate (QF, analogous to QTIE) value is available for inspection or for use in expressions.
2. **CONSTQ** fties extract heat from one lump at a constant rate, and inject it into another lump *irrespective of temperature differences*. The user supplies the heat rate (QF) value. CONSTQ fties can be used *between* a pair of twinned tanks, *but cannot themselves be twinned*.
3. The **AXIAL** ftie is an aid in simulating axial conduction along a duct (within the fluid itself). The user assigns a tube or Stube to the ftie, and the properties of that path (flow regime, shape: TLEN, AF, AFI, AFJ) and of the attached lumps (fluid conductivity, void fraction) are used to automatically compute the conductance (GF) of the AXIAL ftie. This is the same option invoked when the FTIE=AXIAL option is used when generating a tube or Stube (see "Tube and STUBE Input" on page 5-44).

Twin Ftie: The Twin Ftie check box allows the user to specify that the ftie be twinned. CONSTQ fties cannot be twinned. AXIAL fties *must* be twinned if they are used along with twinned tanks and twinned paths. If *either* of the endpoint lumps is twinned, and the tube or STUBE connector is also twinned, then the ftie must also be twinned. However, if the tube or STUBE is twinned and the endpoint lumps are not, then a single (untwinned) ftie may be used. Or, if

the endpoint lump is twinned but a homogeneous path connects just to one of the tanks, then the ftie similarly attaches to the same tank and need not be twinned.

> Note: These rules are obeyed automatically by the "Generate Axial Ftie" option on a tube or a Stube path (see "Tube and STUBE Input" on page 5-44). If the tube or Stube is twinned and attaches to one or more twinned tanks, then the ftie will automatically be twinned (using the same ID as the tube or Stube twin). Likewise, if neither lump is twinned, then the ftie will not be twinned.

Twin Number: This is the twin ftie ID number. This integer is generated automatically based on available IDs. However, the ID can be overridden by the user.

Duplication Factors (DUPC and DUPD): Fties may be assigned duplication factors, which is a general device for exploiting symmetry to reduce the size of the model. For independently generated AXIAL fties, the DUPC and DUPD ftie factors must be the same as the corresponding path DUPI and DUPJ factors (DUPC = DUPI *or* DUPJ, depending on the order in which they were input). This rule is satisfied automatically for AXIAL fties generated in the definition of the path.

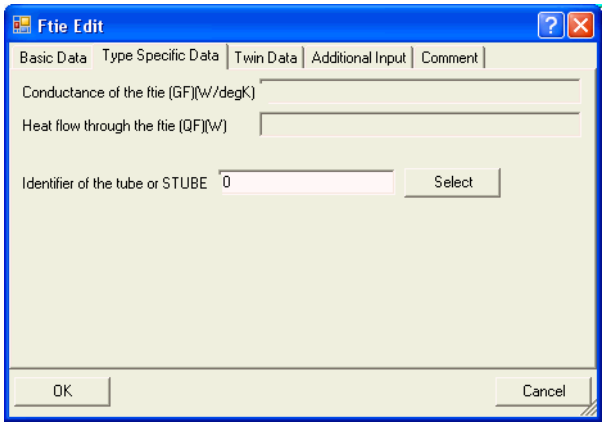
Type Specific Data: The content of the Type Specific tab will vary depending on the type of ftie selected on the Basic tab. Depending on the type of ftie, the User Defined Conductance, allows the user to supply a conductance that will use the temperatures of the lumps to compute the heat transfer. The Constant Heat Flux type allows the user to specify the rate of heat transfer. The Conductance Along Tube (AXIAL) type uses the parameter of a (S)tube to compute the heat transfer based on the properties of the path. A path must be selected for this type; the path box allows the user to change or replace the currently selected path.

Conductance of the ftie (GF) - Conductance of the User ftie. The conductance must be positive or zero. This parameter is ignored for ConstQ fties. For Axial fties, this value is calculated automatically from the path and lump parameters.

Heat flow through the ftie (QF) - Heat flow through the ConstQ ftie (positive from lump C to lump D). This parameter is available as an output parameter for User and Axial fties.

Identifier of the tube or STUBE - This identifies the path associated with the Axial ftie for calculation of the conductance. The Select button may be used to allow path selection in the network window.

FIGURE 5-43. Ftie Type Specific Data

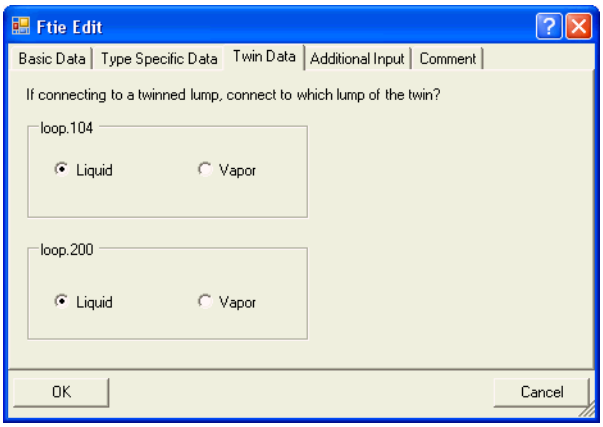


5.4.7.3 Twin Data

When an iface is *not* twinned but is connected to one or two twinned tanks, the user must identify to which side of the twinned tank the iface should connect. This can be done using the Twin Data tab shown in Figure 5-44. The form allows the user to select either the liquid or the vapor side of the twinned tanks.

If the adjacent tank is not twinned the liquid/vapor options will be grayed out and a check box will be provided to twin the lump. Checking this box will twin the tank along with enabling the associated liquid/vapor dialog. If the adjacent lump is a junction or a plenum, the Twin Data dialog will be grayed out.

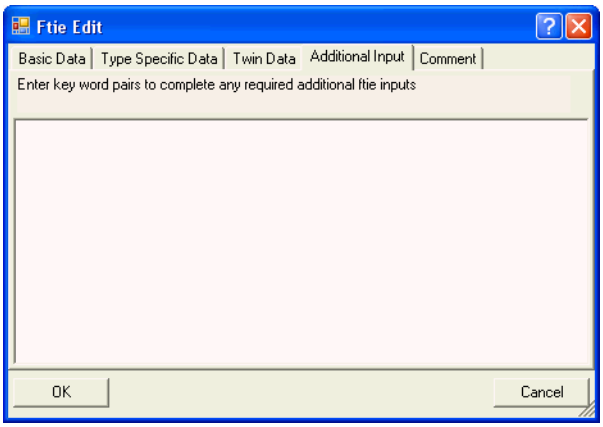
FIGURE 5-44. Ftie Twin Data



5.4.7.4 Additional Input

The Additional Input tab provides a means for the user to define less commonly used or advanced parameters not available to the user on the Type Specific Data tab. This tab is simply a text input window allow the user to define additional parameters in the standard ASCII text format as defined in the SINDA/FLUINT User's Manual. Since these inputs are column sensitive and should not start in column 1, Sinaps will automatically start the inputs in column 2. Therefore comments cannot be designated using a 'C' in column 1. However in-line '$' comments can be used.

FIGURE 5-45. Ftie Additional Input



5.4.7.5 To Move and FTIES

Fties can be relocated simply by clicking on the ftie link to highlight it and make the grip points visible. Then, drag the appropriate grip point from an existing lump to a new lump in the same submodel.

5.4.8 Interface (IFACE)

When lumps are interconnected by paths, mass and energy flow between the lumps according to pressure drops and other forces. Tanks (and only tanks) may also be linked together using ***interface*** elements (interfaces or “ifaces” for short). Unlike a path, no mass flows through an iface. Rather, an iface describes the wall between the two tanks and how it behaves. It tells the program that two adjacent control volumes (tanks) share a common boundary, and describes how their pressures and volumes are interrelated.

For example, consider the simplest and most common iface: a FLAT iface. Placing a FLAT iface between two tanks means that the two tanks share an infinitely flexible boundary. Pressure differences between the two tanks will be essentially zero, since any change in pressure in one tank would cause the interface to shift: one tank will grow and the other will shrink until pressures are equalized. Implicit in this behavior is that the volumetric growth of one tank is equaled by the shrinkage of the other: the volume of the pair of tanks is conserved.

In other words, if only a path (such as a LOSS connector) had been placed between the two tanks, then mass would have moved from one to another until the pressures were equal, but the control volume boundaries would not have changed. Instead, when a FLAT iface is placed between them, the boundary moves but no mass crosses the interface. Of course, ifaces and paths can be used concurrently.

By default, the volumes of either tank may grow arbitrarily large or small. However, the user may also set an upper and/or lower limit on the volumes. When the upper or lower limit is reached, the iface is deactivated until pressure forces or other changes return it to within the active range.

Sinaps provides six types of ifaces: FLAT, OFFSET, WICK, SPHERE, SPRING, and NULL. For full details on ifaces types, available input parameters, pressure/volume relationships, mass effects, customization, etc., please refer to Section 3.8 in the SINDA/FLUINT User's Manual.

> Note: Ifaces can only be connected to lumps which are defined as tanks. Since tanks are treated as junctions in FASTIC ("STEADY") solutions, ifaces do not exist in FASTIC and will be ignored.

5.4.8.1 To Create an Iface

When creating ifaces in the network diagram, there are two methods as described below. Ifaces can only be created between lumps which are defined as tanks. By default, Sinaps will create a Flat iface, although this can be changed through editing. After being created, all ifaces must be edit to set the required input parameters.

1. Fluid Icon Toolbar - Click on the iface icon ( ) in the tool bar and then draw the link between two tanks and the iface will be created.
2. Tool Pallet - Select the iface device on the Fluid>Other Tool Pallet to be created, then draw a link between two tanks, and the iface will be created. Alternatively, drag the iface icon onto the network and then click on the end grip points and move them to the appropriate tanks to connect.

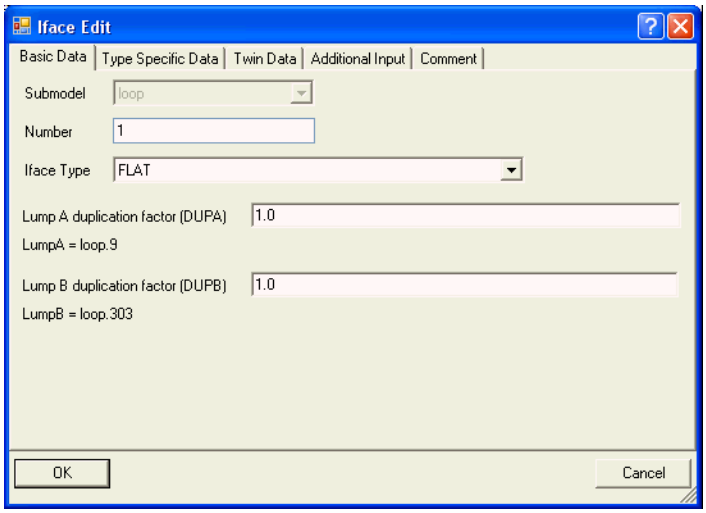
When creating an iface in Sinaps, the first lump selected will be the reference tank, referred to as "tank A." The second lump will be "tank B." These designations affect volume limits and duplication factors as well as other options that depend on the iface type.

5.4.8.2 To Edit an Iface

A iface can be edited using either the Object Properties window or the Iface Edit form. The Iface Edit form can be accessed by double-clicking on the iface, or by right-clicking on a iface to and selecting Edit Iface. Alternatively, clicking on a iface will highlight it (grip points will become visible). Once selected, the iface data is displayed in the Object Properties window and can be edited directly.

The Iface Edit form is a tabbed form providing access to various input options as shown in Figure 5-39. The Basic Data tab provides access to define the iface identification number, type of iface, and duplication factors. The ID must be a positive integer value.

FIGURE 5-46. Iface Edit Form



Iface Type or Classification: There are six types of ifaces available to the user, as described below.

FLAT - This is the most common type of iface. The FLAT iface is massless. There should be no pressure difference between tanks, other than small errors caused by computational approximations and perhaps by hydrostatic (body force) terms. The pressure relationship for a FLAT iface is therefore $PL_a=PL_b$. FLAT ifaces are useful for modeling liquid/vapor interfaces with negligible curvature (negligible surface tension forces), very flexible membranes, frictionless pistons, or even as general arbitrary boundary between two control volumes: FLAT ifaces may be used to arbitrarily subdivide a control volume in one or two or three dimensions. They are applicable for modeling thermal stratification in an otherwise stagnant tank, as an example.

OFFSET- An OFFSET iface is similar to a FLAT interface, but allows the user to specify a pressure difference, DPAB, between tanks. The pressure relationship for an OFFSET iface is therefore $PL_a=PL_b+DPAB_i$. OFFSET ifaces are useful for modeling vertical pistons with weight, the weight of liquid slugs in a small diameter line, capillarity within a subsaturated wick, etc.

SPRING - A SPRING iface is similar to an OFFSET interface, but instead of a constant offset term DPAB, the pressure difference between tanks (in addition to those caused perhaps by body force terms) follows a spring relationship. In a linear spring, the force is equal to a displacement from the equilibrium (zero force) position: $F = k(X-X_o)$. Dividing by cross sectional area squared yields a pressure/volume relationship: $\triangle PL = k'(VOL-VOL_o)$ where $k' = k/A^2 = DPDV$. Other uses for SPRING ifaces include modeling bellows accumulators, elastomeric bladders, meniscus spring action, and certain servo-controlled valves.

WICK - A WICK iface is principally used to model a liquid-vapor interface within or near a capillary structure such as a porous material. It is often used in parallel with CAPIL connectors (Section 5.4.5.6) and CAPPMP macrocommands (“Capillary Evaporator Pumps (CAPPMP)” on page 7-1) to more completely model a section of wick, and in fact a Wick iface can be directly linked to those elements. When the wick is flooded, it behaves similar to a FLAT iface since no meniscus exists. When the wick is subsaturated (and hence the liquid/vapor interface is contained within the wick), it behaves similar to an OFFSET iface with the pressure difference between the tanks dominated by the capillary wick-

ing force $2\sigma/r_c$, where σ is the surface tension and r_c is the effective 2D pumping radius of the wick. When the wick is saturated and the meniscus is greater than r_c but is not infinite (the flooded or flat interface state), then the WICK iface is similar to a nonlinear SPRING interface within its active range.

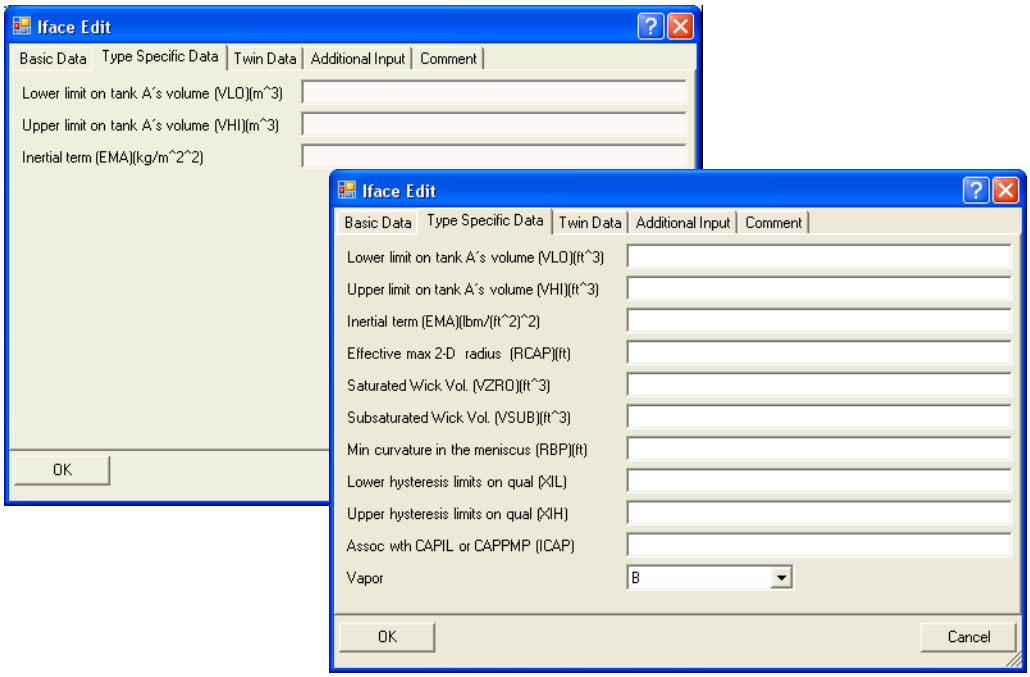
SPHERE - A SPHERE iface is similar to an OFFSET interface, except the pressure difference between tanks (in addition to that caused perhaps by body force terms) is calculated assuming Tank A is a spherical bubble. The pressure relationship for a massless SPHERE iface is therefore $PL_a = PL_b + 2\sigma/r_c$. When using a SPHERE iface, the working fluid must have a surface tension. Any mass transfer across the boundary (e.g., dissolution, condensation, etc.) must be represented by paths. SPHERE ifaces are useful for modeling certain aspects of bubbles. A plurality of bubbles and perhaps distributions of bubbles can be modeled using duplication factors.

NULL - The NULL iface assumes no device type: update of the underlying modeling parameters are left to the user for specialized modeling needs. Because of the sensitivity of model reaction to these values, this option should be considered only by the advanced user.

Duplication Factors (DUPA and DUPB): Ifaces may be assigned duplication factors, which is a general device for exploiting symmetry to reduce the size of the model. The key to understanding duplication factors is this: *tank A "sees" a iface DUPA times, while tank B "sees" the same iface DUPB times.* The A and B duplication factors do not need to be equal. Duplication factors do not need to be integers. Setting DUPA=DUPB=0.0 effectively disables the iface.

Iface Type Specific Data: The input fields of the type specific tab will vary depending on the type of iface selected on the basic tab. Below is a description and guidance for the most common iface input parameters. For full discussion on available input parameters and their usage, please refer to section 3.8 of the SINDA/FLUINT User's Manual.

FIGURE 5-47. Iface Type Specific Data Forms



Lower/Upper Limit on Tank A Volume (VLO/VHI) - Optional lower/upper limits on tank A's volume for the iface to be active (not applicable for NULL ifaces). VLO and VHI default to zero and 1.0E30 respectively. VLO must be less than or equal to VHI. Tank A (the first selected during creation) will be restricted to be less than VHI and/or a minimum value of VLO. If an iface hits one of these limits, and pressure forces are such that it will not move off of it, then the iface is internally disconnected and deactivated. If pressure forces later change such that Tank A moves off the stop (would tend to grow more than VLO or less than VHI), the iface is automatically reactivated by the code. For spring ifaces, VLO and VHI may be used to represent mechanical stops.

Inertial Term (EMA) - If needed, the inertia of the boundary may be modeled via the EMA parameter. EMA defaults to zero. Otherwise, it may be specified for any iface as the mass of the boundary divided by its surface area squared. Hence, in standard FLUINT units EMA has units of either kg/m^4 or lb_m/ft^4. For example, to model a piston of mass M and cross sectional area A, set EMA = M/A^2. This mass is attributed only to movements resulting from changes in the volume of Tank A (the reference lump for the iface).

Pressure Difference Between Tanks (DPAB) - Applicable only to OFFSET ifaces. The user must specify a pressure difference, DPAB, between tanks. The pressure relationship for an OFFSET iface is therefore $PL_a=PL_b+DPAB$. Setting DBAB=5.0 results in Tank A being 5 units of pressure higher than Tank B. DPAB defaults to zero. An OFFSET with DPAB=0.0 is identical to a FLAT iface.

Spring Constant (DPDV) - Applicable only to SPRING ifaces. DPDV is the volumetric spring constant in the pressure relationship $PL_a=PL_b+DPDV_i*(VOL_a-VZRO_i)$. The spring constant can be any value, and has units of force/length^5. The default setting, DPDV=0.0, makes a spring iface identical to a FLAT iface.

Tank A Volume at Zero dP or Saturated Wick Volume (VZRO) - Applicable only to SPRING and WICK ifaces.

> **Spring usage:** *Tank A Volume at Zero dP* - This is the volume of tank A at which point there is no pressure drop between it and Tank B. VZRO can be any value, including negative. Along with DPDV, it simply determines the spring force line in point/slope form.
>
> **Wick usage:** *Saturated Wick Volume* - VZRO defines the volume of the tank A where the wick is fully saturated and the meniscus is absent (the interface is flat). Unlike a spring iface, VZRO therefore has physical meaning and cannot be zero or negative, and in fact it must be greater than VSUB. There is no default value for VZRO. If Tank A is the designated liquid lump, then VZRO must exceed VSUB. Assuming that tank A is liquid, if the volume of tank A is greater than VZRO, then the pressure relationship is a function of the backpressure meniscus (RBP):
>
> $$\Delta PL_a - \Delta PL_b = (PL_{b,\,current} - PL_{a,\,current}) - 2\sigma / RBP_i$$

Subsaturated Wick Volume (VSUB) - Applicable for WICK ifaces only. This is the maximum volume of tank A where the meniscus is assumed to be maximum and constant (unlike real wicks, which exhibit hysteresis and liquid within them does not recede or fill as a flat front). If tank A is the liquid tank, VSUB must be less than VZRO. If the volume of tank A is less than VSUB the pressure relationship is a function of the capillary radius (RCAP):

$$\Delta PL_a - \Delta PL_b = (PL_{b,\,current} - PL_{a,\,current}) - 2\sigma / RCAP_i$$

The above relationships assume the default configuration with Tank B representing the vapor side of the wick. If instead VAPOR=A is specified, then VSUB must exceed VZRO since they are now defined relative to the vapor side of the wick instead of the liquid side.

Effective maximum 2-D radius (RCAP) - Applicable to WICK ifaces only. The effective maximum two dimensional pumping radius of the wick. If Tank A is the liquid tank, and if the volume of Tank A ever drops below VSUB (the subsaturated volume), the meniscus is assumed to be maximum and constant (unlike real wicks, which exhibit hysteresis and liquid within them does not recede or fill as a flat front), resulting in the following pressure relationship:

$$\Delta PL_a - \Delta PL_b = (PL_{b,\,current} - PL_{a,\,current}) - 2\sigma / RCAP_i$$

Minimum Curvature in the Meniscus (RBP) - Applicable to WICK ifaces only, this the characteristic minimum curvature in the meniscus even while flooded: the “backpressure” meniscus. RBP may be any positive value, but it is normally much larger than RCAP and therefore produces a much smaller pressure difference. It defaults to infinity (1.0E30), which means zero pressure difference between liquid and vapor in a flooded state. While RBP may have physical meaning based on the characteristic dimensions of the vapor space, its main use is to avoid artificial flooding as might be caused if capillary paths are in parallel to the iface and the liquid tank is subcooled. For this usage, RBP should be chosen to create a backpressure greater than the error of the solution: about 0.01 psid or 70 Pa.

Lower/Upper Hysteresis Limits on Quality (XIL and XIH) - Applicable to WICK and SPHERE ifaces only, these limits describe the upper and lower hysteresis band limits on quality for the iface to be active. They inactivate the iface if the vapor tank is too wet or if the liquid tank is too dry for a meniscus to be present. If the quality of the liquid tank (Tank A by default) ever exceeds XIH or if the quality of the vapor tank (Tank B by default) ever drops below XIL, the iface is inactivated. To be reactivated, the quality of the liquid tank must drop below XIL and the quality of the vapor tank must exceed XIH.

Association with a CAPIL connector or CAPPMP (ICAP) - Applicable to WICK ifaces only. A common usage is to place a wick iface in parallel with a CAPIL connector or a CAPPMP macrocommand. A CAPIL (Section 5.4.5.6) represents the mass flow through the wick, perhaps zero if a meniscus is present. A CAPPMP macro (“Capillary Evaporator Pumps (CAPPMP)” on page 7-1) performs this feat and perhaps capillary pumping (vaporization at the meniscus) as well. (In such models, RBP should be specified to prevent artificial flooding of the vapor tank due to the assumption of perfect mixing within it.)

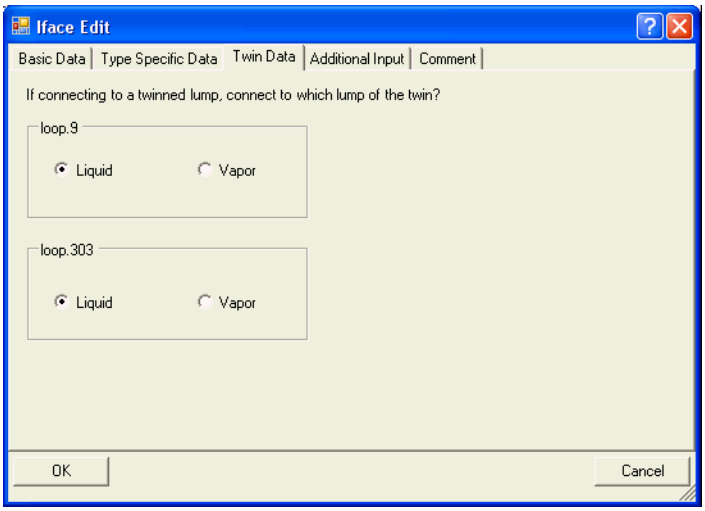
Vapor Tank (VAPOR) - Applicable for WICK ifaces only. Tank A, the first tank selected when creating an iface, is used as a reference. By default, Tank A should contain liquid and Tank B should contain vapor for the wick to be active. For Tank A to be the designated vapor lump, the user should instead specify “VAPOR=A” in the definition of the WICK iface, otherwise VAPOR=B is the default.

5.4.8.3 Twin Data

When an iface is *not* twinned, but is connected to one or two twinned tanks, the user must identify to which side of the twinned tank the iface should connect. This can be done using the Twin Data tab shown in Figure 5-48. The form allows the user to select either the liquid or the vapor side of the twinned tanks.

If the adjacent tank is not twinned the liquid/vapor options will be grayed out and a check box will be provided to twin the lump. Checking this box will twin the tank along with enabling the associated liquid/vapor dialog. If the adjacent lump is a junction or a plenum, the Twin Data dialog will be grayed out.

FIGURE 5-48. Specifying Iface Connections to a Twinned Lump



5.4.8.4 Additional Input

The Additional Input tab provides a means for the user to define less commonly used or advanced parameters not available to the user on the Type Specific Data tab. This tab is simply a text input window allow the user to define additional parameters in the standard ASCII text format as defined in the SINDA/FLUINT User's Manual. Since these inputs are column sensitive and should not start in column 1, Sinaps will automatically start the inputs in column 2. Therefore comments cannot be designated using a 'C' in column 1. However in-line '$' comments can be used.

5.4.8.5 To Move an Iface

Interfaces can be relocated simply by clicking on the iface link to highlight it and make the grip points visible. Drag the appropriate grip point from the existing lump to a new lump in the same submodel.

5.5 User-defined Data Tables

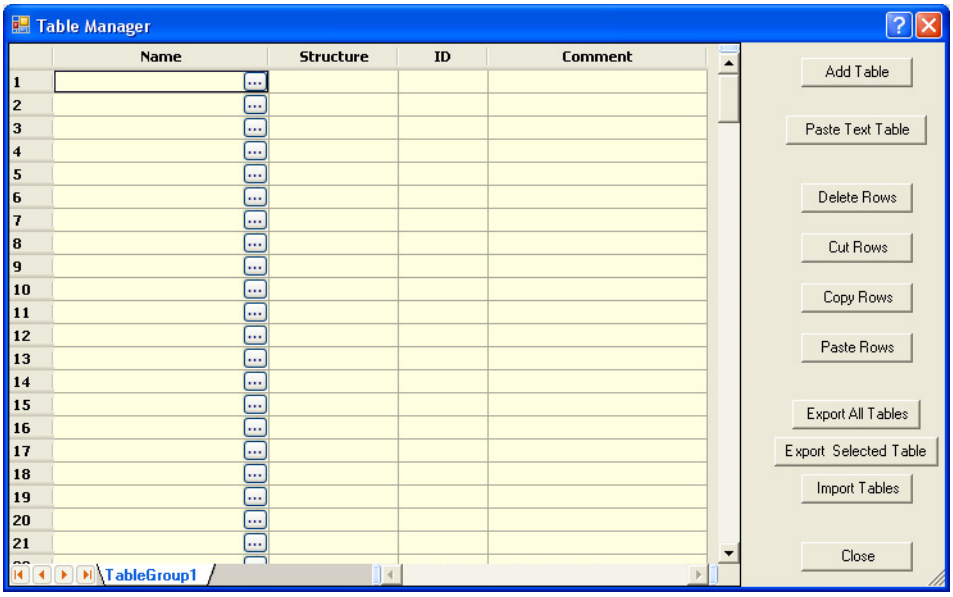
In addition to the network and its parametric definitions, the user will often need to supply additional supporting data, often in the form of arrays or tables of data. Sinaps ***Tables*** provide a user friendly means of defining SINDA/FLUINT arrays or any other type of columnated or tabulated data. A table is owned by a particular submodel, whether thermal or fluid, even though they can be accessed and used by other submodels. When Sinaps writes out the SINDA/FLUINT input file, tables are converted to SINDA/FLUINT arrays. The SINDA/FLUINT User's Manual, Section 2.11 provides additional information on arrays and their usage.

5.5.1 Table Manager

Access to this Sinaps Table Manager dialog is provided through the model browser or by clicking the Table icon ( ) in the tool bar. Many objects such as SIV nodes and turbomachinery devices provide direct access to Tables from the

edit forms. Case level overrides of a table can be defined through the Case Manager (see “Using the Case Manager” on page 9-1).

FIGURE 5-49. Table Manager



The Table Manger lists all user defined tables and some of their key properties such as name, structure, ID, and a comment or description. These properties are only editable within the Table Edit dialog, not through the Table Manager. Similar to the Registers dialog, additional tabs can be added to the Table Manager to organized table perhaps by function or by submodel. To add a new tab, right click on the existing tab and select Add Tab. Tabs can be renamed, deleted, and reorganized.

Once a table has been created, the buttons on the right side of the Table Manager allow tables to be cut, copied, pasted, and deleted. To move a table from tab to a new tab, it must be cut and pasted. When using the cut, copy, paste and delete options, select a table by clicking on the row number to highlight it, then click the action to be performed. Note the traditional Windows short cuts i.e. delete key, ctrl-x, ctrl-c, and ctrl-v, etc. will not work properly in the table manager dialog.

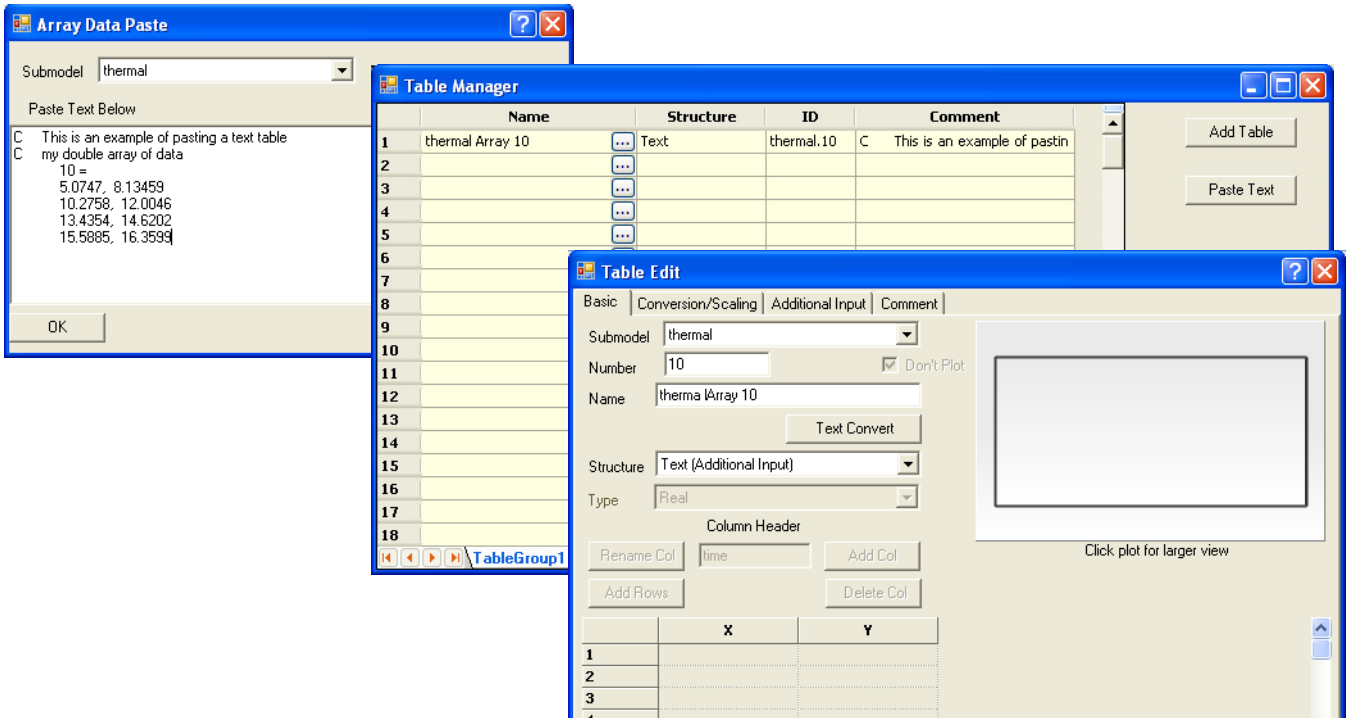
Additional buttons provide the ability to export and import tables between model files (see “Importing and Exporting Tables” on page 5-128).

5.5.2 Paste Text Table

The Paste Text Table button allows the user to quickly paste a text-based SINDA/FLUINT array into the table manager. Clicking the “Paste Text Table” button brings up the Array Data Paste dialog shown on the left below. In this window you must select which submodel where the table will reside and then paste the text into the text window. Sinaps will stip off the array ID, any preceding comments, and create a text based table with the array data.

Note this feature can be used to paste an entire array text block. Simply copy the contents of an array block (without the header line), paste into the Array Data Paste form, select the submodel where the arrays should be placed, and click OK. This will result in all arrays in the block being pasted as multiple tables.

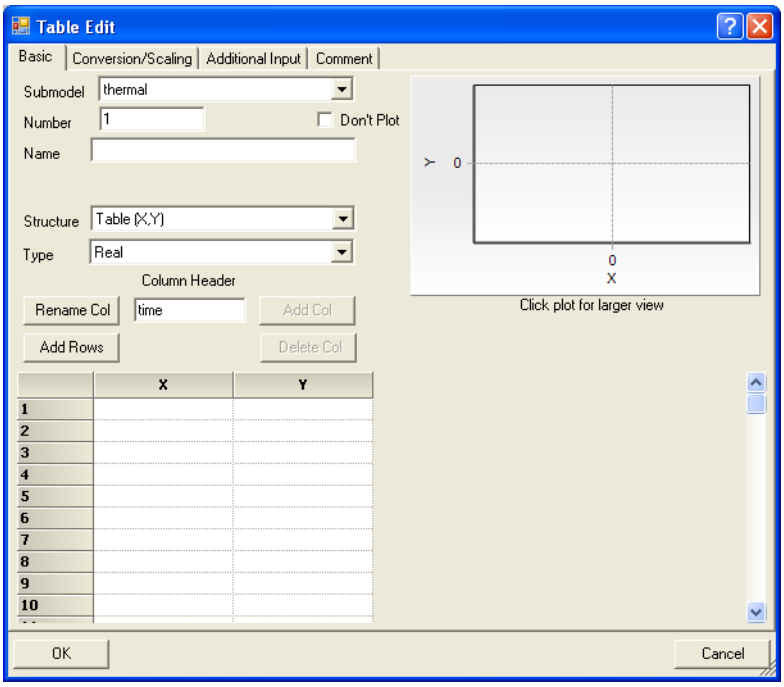
FIGURE 5-50. Pasting a SINDA/FLUINT Text Array



5.5.3 *Creating Tables*

To create a table, open the Table Manger (“Table Manager” on page 5-121) and either click the Add Table button or click on the [...] symbol on a line in the table to open the Table Edit dialog shown in Figure 5-51. Alternatively, the Paste Text Table button allows the user to simply paste a SIND/FLUINT text based array input into the dialog as shown in Figure 5-50. Once in the table manager the text array can be left as a text structure table or it can be converted to the appropriate table structure.

FIGURE 5-51. Table Edit Dialog



Submodel - The submodel the table (SINDA/FLUINT array) will reside in.

Number - The identification number for the table. Numbers are automatically assigned by Sinaps but can be overridden by the user. When the SINDA/FLUINT input file is created, the number will be assigned as the array ID.

Name - The name for the table. The name and the number will appear in the table manager for identification purposes. When the SINDA/FLUINT input file is created, the name will be included as a comment in ARRAY DATA as documentation.

Select Override - This option is only viewable if the table edit form is accessed through the Case Manager table override dialog. See “Tables” on page 9-18 for information on case level table overrides.

Structure - The format of the table. Currently four options are available to the user*.

> ***Single Column*** (singlet array created), contains a sequence of data values of the same type representing various values of the same parameter.
>
> ***XY Table*** (doublet array created), contains a sequence of ordered pairs of values usually representing points on a plane curve. Often used with SINDA/FLUINT built-in interpolation options and subroutines.

* A fifth option, a Multi-Y Table, is used internally by Sinaps for defining turbomachinery devices. This format provides multiple Y values for a single X value and passes each column to SINDA/FLUINT as a separate singlet array.

XY map (bivariate array created), used to represent a .function of two independent variables: Z = f(X,Y). See XY Map Table Structure.

Text, uses the Additional Input tab to define an array in traditional SINDA/FLUINT text format.

Type - Designates the data type contained within the table. The type field is not applicable to tables with structure set to "text". Tables with the structure set to "XY map" will automatically have the Type field set to as "mixed".

Real, contains real numbers, registers, expressions, and/or processor variables. If the user inputs an integer number, Sinaps will automatically convert it to a real number. This is the default type setting.

Integer, contains only integer numbers, no registers or expressions are allowed. Note, integer tables cannot be scaled.

Mixed, may contain a mix of integers, reals, registers and expressions. When mixed is selected, the data will be passed along exactly as input in the table and Sinaps will automatically select the Mix Array option in the Options Data block.

Table Input - The is the area of the form the data is defined as real and/or integer values*, registers or expressions. Data can be pasted from Excel or Word using the standard Window's copy and paste commands (control-c and control-v). The number of columns in the table and the table layout will vary depending on the table structure selected. Columns and rows may be added to the table if necessary through the Add Rows/Col buttons. Likewise columns can be deleted using the Delete Col button.

Column names are used as axis titles in the data plot. The columns can be renamed by clicking on the column title, typing a new name in the Column Header field, and then clicking the Rename Col button.

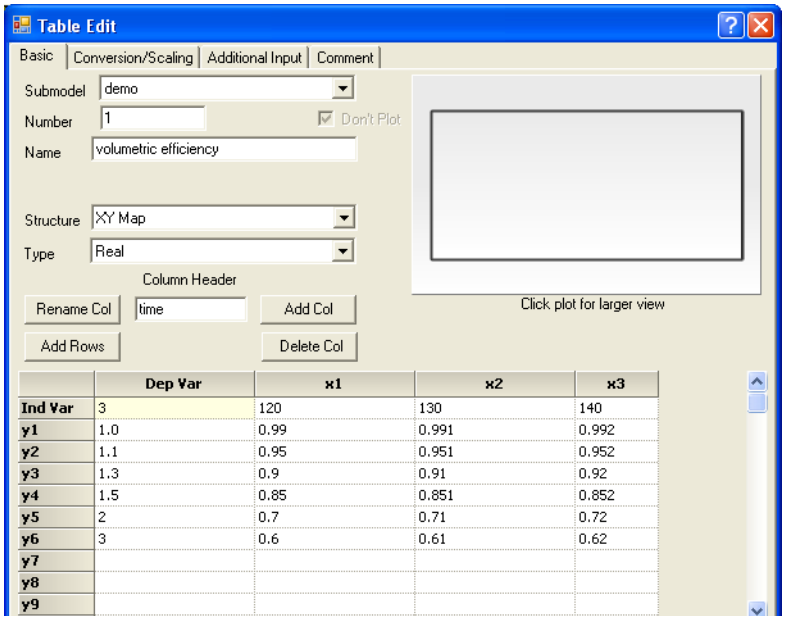
Data Plot - For Single Column and XY Table structures, if at least two rows of data are defined a plot of the data will be displayed allowing visual verification of correct data entry. Clicking on the plot will bring the plot up in EZXY providing a larger view. The plot may be turned off by clicking the Don't Plot check box. Note, the data plot option is disabled for XY map tables.

XY Map Table Structure. When using the XY map option the first row contains the first independent variable values (X_1,X_2,X_3,...), the first column contains the second independent variable values (Y_1,Y_2,Y_3,...), and the interior of the table contains the dependent variable values (Z_{11}, Z_{12}, Z_{13},Z_{21},...) where Z = f(X,Y). The cell in column 1, row 1, contains the total count of the number of X values ("*n*"). This cell is defined by Sinaps and not editable by the user.

* Because the data is entered and preprocessed before SINDA/FLUINT knows the use to which each array is going to be applied, arrays represent a special challenge to the user: extra care should be taken in their entry to avoid late-detected or even undetected problems in the data.

$$
\begin{array}{|l|l|}
\hline
n, & X_1, X_2, \ldots, X_n \\
\hline
Y_1, & Z_{11}, Z_{12}, \ldots, Z_{1n} \\
Y_2, & Z_{21}, Z_{22}, \ldots, Z_{2n} \\
\vdots & \vdots \\
Y_m, & Z_{m1}, Z_{m2}, \ldots, Z_{mn} \\
\hline
\end{array}
$$

An example of a XY map table is shown below.

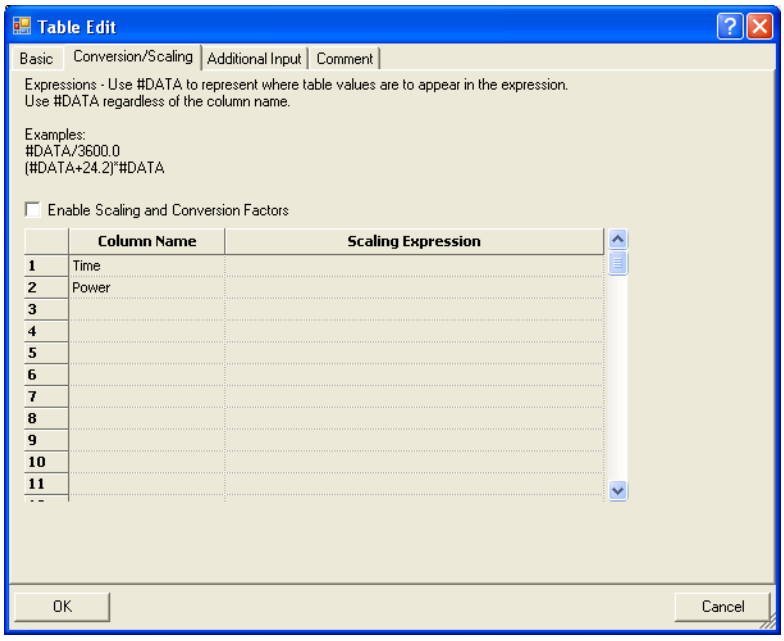


5.5.3.1 Conversion and Scaling

When defining data within tables, scaling factors can be applied through the Conversion/Scaling tab. Such factors may be used to model uncertainty in the data, to scale the data, or to support parametric sweeps on the data. To enable scaling, the Enable Scaling and Conversion Factors box must be checked which will activate the subsequent grid input. Scaling factors are not applicable to talbes where the Type field is set to “Integer”.

Each column in the table is represented by a row in the grid. When creating expressions, use #DATA to represent the data from the table. For example if row 2 is power, you can convert data in Watts to BTU/hr by using #DATA*3.412 or by using the built-in register #data/btuhrwat. Clicking the middle mouse button in any scaling expression field will bring up the register options menu.

Note that when conversions and scaling factors are applied in a table, the plot contained within the table dialog will reflect the scaled data. If the expression cannot be resolved, perhaps due to the use of processor variables, clicking the Don't Plot box on the Basic tab of the table dialog will disable plotting.



5.5.3.2 Additional Text

This tab allows the user to define a traditional text input for a table. The input format for a text table is the standard SINDA/FLUINT array format less the ID:

dv_1, dv_2, dv_3 ...
dv_{n-1}, dv_n

where dv_1, dv_2, ... are data values contained in that array. Lines may wrap, but avoid any input in column 1. Each text table is limited to defining a single SINDA/FLUINT array.

This option exists primarily to support migration of older models and reading in text based SINDA/FLUINT files. It also allows the user to define less common array structures which may not be support by Sinaps Tables such as the tri-variate option. When older models (prior to the existence of Sinaps Tables) are opened or a SINDA/FLUINT text file is read in, all arrays will be migrated as Text structured arrays and the text will be placed on the Additional Input tab. The user may subsequently convert these text arrays to tables using the Convert Text button on the table form.

5.5.3.3 Table Comment

The Comment tab in the Table edit form may be used to include a description of the table or other documentation such as references etc. When text based arrays are imported into Sinaps, any comment lines starting with a 'C' located before an array description will be imported into the Comment tab.

5.5.3.4 Importing and Exporting Tables

Using the Table Manager, tables can be exported or importing allowing the transfer of table between models or perhaps between cases. To export a single or select group of tables, highlight the table rows in the Table Manager and click on the Export Selected Tables button. To export all tables, use the Export All Tables button. Exported tables will be stored in a *.xml file which the user will be prompted to define during the export process.

To import tables, click on the Import Tables button and browse to the [table].xml file. In the exported tables file, each table belongs to a submodel. When importing tables, they will be placed in the submodel with the same name by default, assuming you have such a submodel defined. If the original submodel (owner of table) is not found, the tables will be placed in the current thermal submodel. Tables will be automatically renumbered during import.

Sinaps will also check for duplicate tables names during import. If duplicate names are found the user will be prompted to either overwrite the existing tables or add new tables. The selected action (overwrite or add) will apply to *all* tables being imported. Sinaps will add a suffix to the name of any new tables created.

5.5.4 Referencing Tables (or Array Data) in Logic or Expressions

Since tables are an interface for defining SINDA/FLUINT arrays, the data contained in the tables can be referenced in other data fields, expressions or user logic. To aide in the process, the Table Manager dialog can be accessed directly from the Case Manager Logic Blocks and from the Network Logic tab on any network elemement by simply right clicking in the text editor and selecting *Edit Tables*. This allows tables to be viewed, added, or edited without the need to exit the text editor or network logic edit dialog.

In logic or expressions, the user can refer to data cells within arrays using the following formats. Note the knowledge of the order in which the table is written out to the input file is required.

```
sub.A(n+i) .  ith value of array in submodel smn whose ID is n, real value
sub.NA(n+i)   ith value of array in submodel smn whose ID is n, integer value
```

For example, the following refers to the third (real) value in array #405 of submodel *hotrod*:

```
hotrod.A(405+3)
```

5.6 User-defined Logic

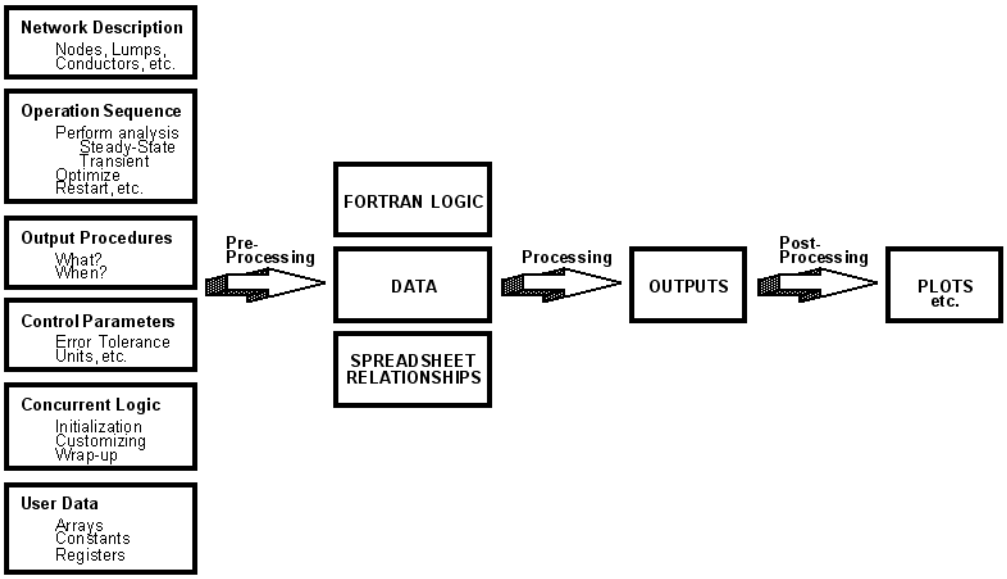
"Parameterizing a Model" on page 6-1 demonstrates how many model manipulations can be accomplished using the parametric system based on registers and their expressions, which might include processor variables (e.g., the current problem time, the temperature of a particular lump, etc.). However, there is a limit to what can be accomplished in such spreadsheet-like system. For example, the user may wish to call a library routine for a particular heat transfer correlation, or even write their own correlation. Therefore, a concurrent system exists that permits extensive customization: user-defined logic.

While concurrently-executed logic can be considered an advanced option, many basic options are performed by such blocks, including the analysis sequence (Operations), and the output commands (Output Calls). Sinaps assists in the

automatic generation of logic for the most common choices, usually via options in the Case Manager (see "Using the Case Manager" on page 9-1, and Section 9.3.5.13 in particular). The intermediate user will soon find the need to use their own custom sequences, including access to registers and the spreadsheet system, and the advanced user will find that the logic blocks provide unlimited* access and modeling power.

SINDA/FLUINT consists of a preprocessor executable program and a processor library of pre-compiled routines (including everything from the STEADY solution routine to the LMPTAB output routine). The preprocessor scans the input file, a procedure that includes the creation of a custom Fortran program (along with the data that it will use). This Fortran program (named *astap.for*) contains automatically generated logic as well as direct and translated user logic. It is passed to a Fortran compiler (whether external or built-in), which links with the processor library to create a custom executable program every time a case is run. The data flow for this three-step process (the third step being post-processing) is depicted in Figure 5-52.

FIGURE 5-52. Simplified SINDA/FLUINT Data Flow



5.6.1 Fortran and Compiling

Logic blocks are distinguished from data blocks in that logic blocks define a sequential list of operations (updating, outputting, solving, etc.) to be performed: it matters *when* during the solution that the particular logic block is invoked (Section 5.6.5, Section 5.6.6), and the *order* in which the commands within that block are invoked. Furthermore, SINDA/FLUINT is largely oblivious to the exact content of those instructions: it searches through them for content it understands, and it responds to commands *as they are executed*, but it has little knowledge beyond that and so it cannot detect the user's intent nor trap certain mistakes.

* SINDA/FLUINT can literally be rewritten from within its own logic blocks. At some point, the software's ability to trap mistakes disappears and the advanced user must take an increasing amount of responsibility for well crafted user logic.

Logic blocks are scanned by SINDA/FLUINT's preprocessor, which looks for variables that it understands such as "rhg.TL321," which means the temperature of lump #321 in fluid submodel *rhg* (see "Referring to Processor Variables" on page 5-133). If these types of variables are detected, SINDA/FLUINT will convert those instances into internal array references that the Fortran compiler can understand; this conversion is termed *translation*. By default, it will err off if it detects variables or functions that it does not understand, in case those are spelling mistakes.

Both translation and spell checking can be turned off by placing an F in column 1, or by bracketing a section of code by FSTART and FSTOP (which also must be placed in column 1). Translation on any one variable can be turned off by using the fake submodel name "global." Turning off translation is an advanced option, and should be used only when needed. An example of disabled translation is given in Section 5.6.3.

To turn off the spell check feature (for example, to be able to use temporary local variables that are not declared as registers), logic can be bracketed with SPELLOFF/SPELLON commands, which must start in column 1. However, this usage is discouraged as it can result in errors that are difficult to detect. Note that temporary (but global in scope) variables are available to help avoid the need to turn off the spell checker. The temporary variables (technically, "user constants") are named *atest, btest, ctest, ... ztest*. Note that they follow Fortran implicit naming conventions: *itest, jtest, ktest, ltest, mtest, and ntest* are of type integer, while the rest of the temporary variables are of type real (floating point). Double precision user constants (*atest_dp* through *htest_dp*, and *otest_dp* through *ztest_dp*) are also available.

One common mistake is to use the wrong type of number or value as an argument in a subroutine call. For example, if a routine named SIRD has four arguments, and is documented to expect a string argument, an integer argument, and a real argument, and a double precision real argument. The following example applies:

```
call sird('mymod',  10, 0.001, 3.0d4)            $ valid call
call sird(mymod, realReg, 0001, otherReg)         $ four major errors
```

In the second call, a character string has not been supplied, and (if the spell checker was turned off) a zero-valued integer named *mymod* was supplied to the subroutine instead. In the second argument, a real register (recalling that they are real by default) has been supplied instead of an integer, which will send nonsense bits to the subroutine that it will faithfully interpret as a valid integer (perhaps something like "-103892003"). In the third argument, the reverse has happened: by accidently omitting the decimal point, an integer has been passed to SIRD, again creating a nonsense real number such as "1.2934e-44."* In the forth argument, only the first 32 bits of a real (32 bit) register named otherReg has been send to SIRD, and the next 32 bits needed to represent a 64 bit double precision real value are unpredictable.† The exact effect of such mistakes will vary depending on the routine being called, and can range from subtle to catastrophic.

For subroutines in the SINDA/FLUINT library, the compiler will check argument number and type. User-defined subroutines ("User-defined Subroutines" on page 5-136) do not have such protection, so they must be called carefully.

5.6.2 Specifying Logical Instructions

Anything that is valid in an Fortran 95 compiler can be accomplished within a logic block, and these operations become a subset when the additional capabilities of SINDA/FLUINT are superimposed. Clearly, such a scope is too great to be covered in this document. Instead, only a cursory summary of some basic operations will be covered here. The user is

* Fortran is a weakly-typed language that passes arguments by address, not by value.

† Pressures (PL, PLTH) are double precision variables. Double precision registers are also available.

referred to Fortran compiler documentation or texts, and to the SINDA/FLUINT User's Manual (especially Sections 1, 4, and 7) for more details.

5.6.2.1 Column Restrictions

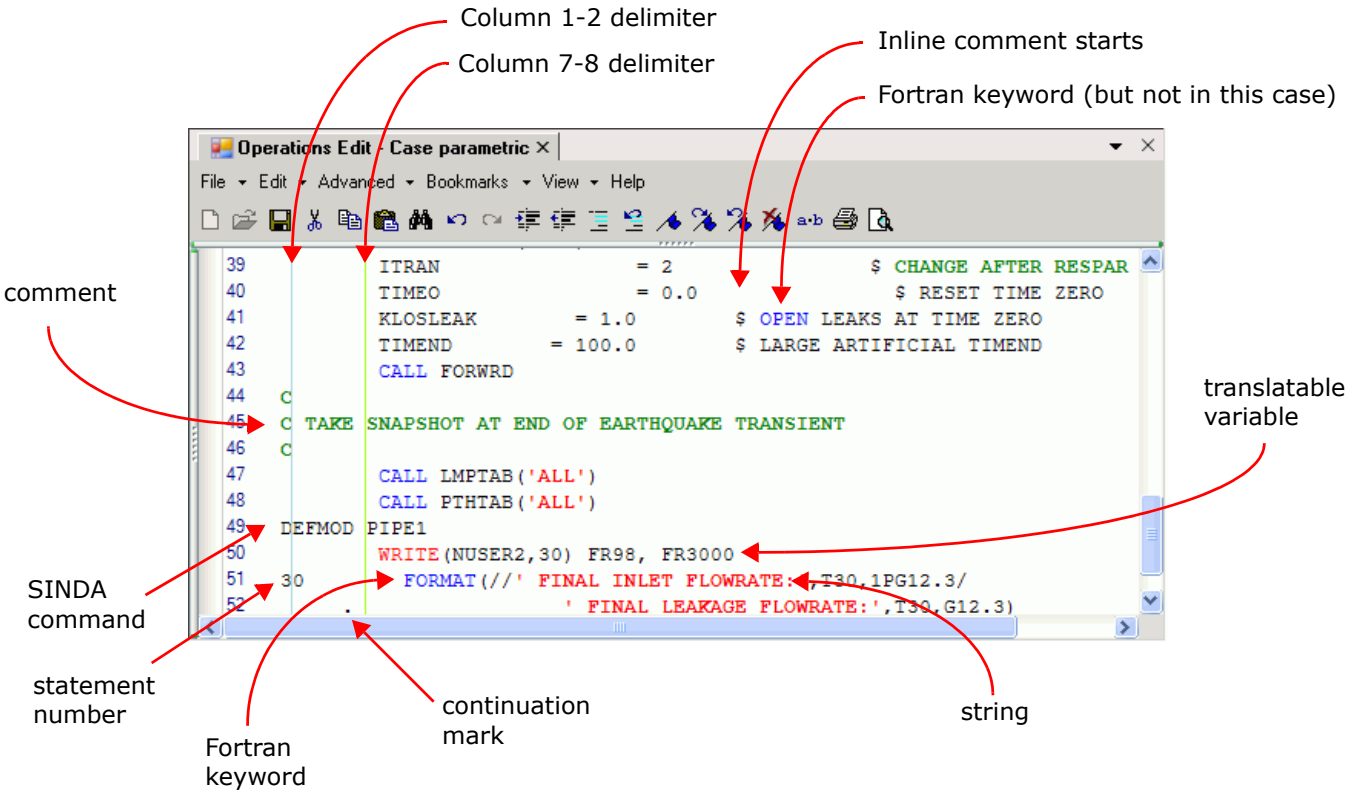
Columns 1 through 6 have special meaning to the Fortran compiler, and no statement can exceed 132 columns. Use a <tab> as the first character to move most statements out of these special columns.

Column 1 is used by SINDA/FLUINT in both data and logic blocks to signal high level commands (such as BUILD and BUILDF, SPELLON/SPELLOFF, FSTART/FSTOP). It is also used by the Fortran compiler to detect a whole-line comment if a C or R is detected in column 1, so be sure the "C" in "CALL ..." is preceded by a tab. (An inline comment can be accomplished using a dollar sign "$".)

Columns 1-5 may be used for Fortran statement numbers, and anything in column 6 denotes that this line is a continuation of the one above it. Therefore, any other statements must begin in column 7 at least.

The Sinaps logic editor marks column 1 with a blue line (between columns 1 and 2) and column 7 with a green line (between columns 7 and 8, with a tab starting text safely in column 9) as an aid in obeying Fortran column conventions. It also attempts to highlight reserved words with special meaning, strings, comments, etc. A sample showing many of these features is depicted in Figure 5-53.

FIGURE 5-53. Example Logic Block in Sinaps



5.6.2.2 Some Simple Statements

Some Fortran statements commonly of use to Sinaps users include setting values ("assignment statements"), calling routines, and writing messages to the output file or to the screen. All of the examples below can be assumed to start beyond column 7.

Assignment statements redefine parameters, such as registers and certain processor variables (although not all of these can or should be reset). The following samples change the values of the register *for_once* and the temperature of the boundary node 999 in submodel *sink* to be 10.0 and *for_once* respectively:

```
for_once = 10.0
sink.T999 = for_once
```

If the above statements had been reversed, then sink.T999 would have been set to the earlier value of *for_once*. Also, note that in the above example, if the temperature of the boundary (or heater) node #999 had been defined to be *for_once* in the input form, then just changing the value of *for_once* would have been enough to change the temperature of the boundary node (automatically at the end of the user logic block, or via user instruction by calling UPREG as described in "Parameterizing a Model" on page 6-1). Note also that the temperature of different types of nodes (arithmetic and diffusion) are normally program outputs and are normally not changed by the user. Finally, note that the temperature of a fluid plenum could not be reset in the above manner; a call to CHGLMP would be required if the value of TL had not be defined to be *for_once* in the input field (in which case an internal call to CHGLMP is generated in the spreadsheet system).

The following statements write the temperature of the above node to the output file (Fortran unit *nout* in SINDA/FLUINT), and another diagnostic message to the screen.

```
write(nout,*) ' The temperature of the boundary is now ', sink.T999,' degrees'
call userwin(' The temperature of the boundary has been reset!')
```

Another common operation is a conditional operation. The follow example changes the value of *for_once* to be either 10.0 or 20.0 depending on whether the current time is past a limit (*tchange*) or not:

```
if(timen .ge. tchange)then
       for_once = 10.0
else
       for_once = 20.0
endif
```

The above could have been accomplished by defining *for_once* in the register table as:

```
(timen >= tchange)? 10 : 20
```

but often the branching can become too complex for expressions, or perhaps subroutines or functions must be called as part of the logic. For example, the integer constant *nsol* can be used in logic or expressions to tell whether the solution is currently in a steady state (*nsol*=0 for STEADY, or *nsol*=1 for the more specialized STDSTL) or a transient (*nsol*=2):

```
if(nsol .le. 1) then
       call userwin(' Currently doing a steady state run.')
else
       call userwin(' Currently doing a transient run.')
endif
```

5.6.2.3 Simple Repetitions

Often, certain subprocedures will need to be repeated several times. Such "Do loops" can take several forms, including one based on a fixed count and another based on a condition remaining true. For example, change the temperature in 9 plena (numbered 101, 201 ... 901) in submodel *flow* to the value *tempNew* holding quality constant:

```
do itest = 101,901,100       $ start, end values with an increment of 100
       call chglmp('flow',itest,'tl',tempNew,'xl')
enddo
```

A "Do while" loop continues until some defined condition is no longer true at the end of the loop. For example, the logic below accomplishes the same as the one above:*

```
itest = 101                     $ initialize
do while (itest .le. 901)
       call chglmp('flow',itest,'tl',tempNew,'xl')
       itest = itest + 100
enddo
```

5.6.3 Referring to Processor Variables

Network element (e.g., node, lump) parameters such as temperature, diameter, etc. are all available in every logic block for inspection (use in calculations, output) or for updating. Control parameters (e.g., simulation end time, accuracy tolerances, current time, current accuracy, etc.) are also available. The list of available parameters runs into the hundreds, and so only a brief summary will be provided here, with full details available in the SINDA/FLUINT User's Manual (with a summary listed in Section 6).

Most of these parameters must be exposed to the SINDA/FLUINT preprocessor (vs. hidden using F, FSTART, FSTOP) in order to be translated into an internal reference that can be compiled. For example, "fsub.DH39" or the equivalent "fsub.DH(39)" ... meaning the hydraulic diameter of tube 39 in fluid submodel *fsub*, might be translated as "DH(3493)" where 3493 is the internal location of that parameter ... a number not normally visible to the user because it is not needed. Review of the generated astap.for will reveal such translations. The user needs to be aware *that* they occur, but normally should not be concerned with the exact details of internal storage rules.

One example of advanced translation will be provided. In the prior section (Section 5.6.2), a "Do loop" changed the temperatures of 9 plena. If instead, the user wished to print out those pressures, it would be cumbersome to write:

```
write(nout,*) 'The temperature at inlet ',101,' is ', flow.tl(101)
write(nout,*) 'The temperature at inlet ',201,' is ', flow.tl(201)
...
```

So instead, the user might choose to loop through the IDs. However, the SINDA/FLUINT preprocessor does not understand the value of itest (since execution has not yet begun) in the following erroneous approach:

```
do itest = 101,901,100
       write(nout,*) 'The temperature at inlet ',itest,' is ', flow.tl(itest)  $ NO!
enddo
```

* Though in this case in a convoluted way that shouldn't be construed as the "Do while" being less efficient, since they are generally not substituted one for the other.

Dynamic translation options are available (see Section 7 of the SINDA/FLUINT User's Manual for a full list) to permit the intent of the above logic. In the above case, the function INTLMP (or subroutine LMPTRN) can be called to perform the translation "on the fly" ... while the SINDA/FLUINT processor executes instead of the normal one-time (hard-wired) translation that would otherwise be performed by the SINDA/FLUINT preprocessor:

```
       do itest = 101,901,100
              ltest = intlmp('flow',itest)       $ dynamically translate itest to ltest
F             write(nout,*) 'The temperature at inlet ',itest,' is ', tl(ltest)  $ YES!
       enddo
```

Note that in the above case, the translation is suspended by placing an F in column 1, which prevents the preprocessor from declaring an error when it encounters "tl(ltest)" and is unable to determine the value of *ltest*.

Network Element Logic is a feature that can help avoid translation, renumbering, and do-loops when the same logic is applied to many network elements at once. See Section 5.6.8.

5.6.4 Interactions with Registers and Expressions

All registers are available in all logic blocks, and are of type real (single precision) unless they have been explicitly checked as integers in the register spreadsheet window. Registers may be inspected or changed. If their value is changed, whether by an assignment statement or as an action of a subroutine, then any expression in which those changed registers have been used will be updated (either at the end of the current logic block, or by explicit force of an update via the user's call to the subroutine UPREG).

Note that some inputs may be defined using registers or register-containing expressions, but changes are ignored during processor execution. These inputs include initial temperatures for diffusion and arithmetic nodes, and initial thermodynamic states for tanks and junctions. Otherwise, the user risks conflict between the solution updates and their commanded updates. This is not true for boundary (and heater) nodes and for plena, since those elements are intended to be boundary conditions set by the user, and are not simulation results set by SINDA/FLUINT. Therefore, the temperatures (and in the case of plena, pressures and qualities) can and often *are* set using expressions, perhaps as functions of time (*timen* being the current time or "time now").

5.6.5 Global Logic Blocks

Some logic blocks occur only once: they are global and not owned by any particular thermal or fluid submodel. The primary example is Operations. ***Operations*** becomes the main routine when SINDA/FLUINT starts: it defines all that SINDA/FLUINT will do in the current session, and when it returns, SINDA/FLUINT execution ends.

The usual statements issued in Operations include BUILD or BUILDF configuration commands, calls to (steady state or transient) solutions, invocation of advanced design modules such as the optimizer or a statistical sampler, and saving or recovering restart sets. For example, the following block builds two thermal submodels, one fluid submodel, does a steady state solution, and saves the result to a binary SAVE file:

```
build config, toohot, toocold
buildf config, justrite
       call steady
       call save('all',0)
```

Other examples of global logic blocks include PROCEDURE and RELPROCEDURE, which are like Operations except that they are called repeatedly by the advanced design modules (see Section 5 of the SINDA/FLUINT User's Manual).

In addition to submodel-level OUTPUT CALLS, VARIABLES, and FLOGIC blocks, global-level blocks are available that are always called independent of any BUILD or BUILDF status, and which always follow the submodel-level blocks during execution. One usage of the global block is to avoid any conflicts updating Q's or T's (e.g., HEATPIPE or TEC calls) by putting those updates in global VARIABLES 1 block. Another usage is to have a single, central place for output operations that is always present in every model.

Within a global-level logic block, there is no presumed submodel: the user must either use the submodel prefix for any processor variables, or they must declare to the preprocessor the current default name. The default submodel name is declared using the DEFMOD command. For example, in Figure 5-53 the DEFMOD command is used to define PIPE1 as the default submodel (fluid, in that case) for the subsequent references to "FR98, FR3000." Without that column 1 declaration, the user would have had to use "PIPE1.FR98, PIPE1.FR3000."

5.6.6 Submodel-specific Logic Blocks

Most of the logic blocks are submodel-specific, including output operations (Output Calls). These blocks will be active if their respective submodel has been activated using a BUILD or BUILDF statement. If the processor variable being used belongs to the current submodel, then the submodel prefix is optional. Otherwise, submodel prefixes should be applied to refer to values in other submodels, and they are always recommended for clarity as well as safety (i.e., in case cutting and pasting operations generate mistakes). These logic blocks are accessible through the advanced tab of the case manager, see "User Logic Blocks" on page 9-33 for additional information.

The Output Calls blocks are called at the start and end of a steady-state solution, and at user-defined intervals (OUTPUT and OUTPTF for thermal and fluid submodels, respectively) during a transient. While they are sensible places for output operations, SINDA/FLUINT does not enforce any restrictions on their use: they may be used for modeling discrete-interval controllers, for example.

Other logic blocks are called more frequently during the generation of the simulation results themselves, are therefore they represent key opportunities for customization.

For thermal submodels, three such blocks are available:

```
Variables 0 . . . Called at the start of a steady-state, or start of a transient time step
Variables 1 . . . Called each steady-state iteration, or during a transient time step
Variables 2 . . . Called at the end of a steady-state, or after a transient time step
```

Notice that a steady state solution is treated equivalently to a single transient time step from the perspective of these logic blocks. Given when they are updated, the intended (but not enforced) usage of the thermal submodel-specific logic blocks is as follows:

```
Variables 0 . . . Time-dependent updates
Variables 1 . . . Temperature-dependent updates
Variables 2 . . . Wrap up before output, or signal completion of a transient (by setting TIMEND=TIMEN)
```

When in doubt as to where to place customization logic, choose Variables 1 for the simple reason that it is invoked the most often. For fluid submodels, a parallel but not completely analogous system exists:

```
Flogic 0 . . . . . Called at the start of each steady-state iteration, or start of a transient time step
Flogic 1 . . . . . Called during each steady-state iteration, or during a transient time step
Flogic 2 . . . . . Called at the end of a steady-state, or after a transient time step
```

Once key difference is that both Flogic 0 and Flogic 1 are called with each solution step, whether a steady-state iteration or a transient time-step. Indeed, when in doubt as to where to place fluid customization logic, *choose Flogic 0* (and not Flogic 1, which is reserved for more advanced operations, perhaps as directed by program documentation).

5.6.7 User-defined Subroutines

If the same operation is to be performed repeatedly in several locations, perhaps it makes sense to define those operations once, as a user-defined subroutine, then call that subroutine from various locations (either within one logic block, or from different blocks). User-defined subroutines can be added via the advanced tab of the case manager, see “Subroutines” on page 9-30 for additional information.

User-defined subroutines generally take two forms: either they are independent of SINDA/FLUINT’s variables, or they depend on them. For an example of the first type, consider a routine that adds one to any argument it receives:

```
fstart
       subroutine add_one(val_to_incr)
       real val_to_incr
       val_to_incr = val_to_incr + 1
       return
       end
fstop
```

In the above example, the FSTART/FSTOP declarations essentially hide the details of this routine from SINDA/FLUINT’s preprocessor, since the operations of this routine have nothing directly to do with SINDA/FLUINT variables (registers, processor variables, etc.). The user might then invoke the above routine within Operations (or some other block) as follows:

```
       plateLen = 1.0
       do while (plateLen .le. 10.0)
              call add_one(plateLen)
              call steady
       enddo
```

Sometimes, however, a user-defined subroutine is an extension of SINDA/FLUINT itself and needs access to its parameters. In the example below, the routine WriteIt issues a statement to the output file:

```
       subroutine WriteIt(lumpNum)
       call common
c
       ltest = intlmp('flow',lumpNum)       $ dynamically translate itest to ltest
F      write(nout,*) 'The temperature at inlet ',lumpNum,' is ', tl(ltest)
       return
       end
```

and it might be invoked as follows (this is functionally identical to an example given in Section 5.6.3).

```
do itest = 101,901,100
       call WriteIt(itest)
enddo
```

Some important differences should be noted in this "SINDA-aware" routine. First, other than one line, the routine has not been hidden from the SINDA/FLUINT preprocessor using FSTART/FSTOP commands, since translatable parameters might be contained within. Therefore, *lumpNum* must be a register to avoid spell checker errors, and in fact must be an *integer* register because of the way it is used. Furthermore, this register, the output unit nout, and the TL array would all be unavailable to that routine without the CALL COMMON command issued in the second line of WriteIt, which causes SINDA/FLUINT to make active all of its internal arrays, variables, etc.

A third possible use for Subroutines is somewhat advanced. By matching the name and arguments of any routine documented in the SINDA/FLUINT User's Manual (Section 7), the user can globally replace that routine. In other words, when the processor is linked, it will link preferentially with routines in Subroutines, thereby ignoring any duplicate routines found in the SINDA/FLUINT processor library. For example, supplying a routine called FRICT would globally replace the "Moody Chart" routine in SINDA/FLUINT with the user's custom wall friction factor:

```
fstart
       real function frict(Reynolds, wall_rough_fract)
       real Reynolds, wall_rough_fract
       frict = ... (user logic, routines, here)
       return
       end
fstart
```

5.6.8 Network Element Logic

Expressions (instead of fixed numbers) inside of Sinaps data input fields provide significant power, but there are limitations. For example, neither user-defined functions nor program library routines can be called, and multiple values cannot be set at once (e.g., both the UB and the UEDT of an HTU tie, whose values are often interrelated).

As a work-around, the user can write logic to be placed in perhaps FLOGIC 0 or VARIABLES 1, but this logic must then reference the nodes, ties, paths, etc. that are defined in the network, making sure that the IDs are correct and have not changed since the last model update.

Network Element Logic provides a means for overcoming most of these limitations. The Logic tabs within the edit forms for nodes, conductors, lumps, paths, ties, fties, and ifaces provide locations for an arbitrary listing of Fortran-based instructions that are associated with each node, conductor, etc. Figure 5-54 shows an example of a node's initialization logic, and Figure 5-55 shows an example of a path's customization logic.

FIGURE 5-54. Example of Thermal Network Element Logic (for a Node)

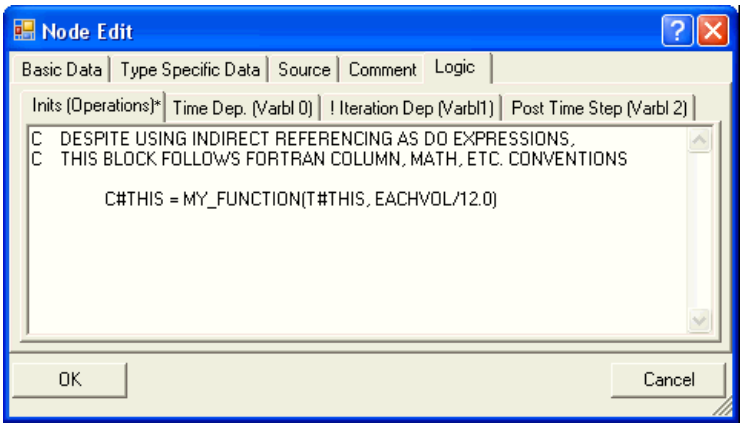
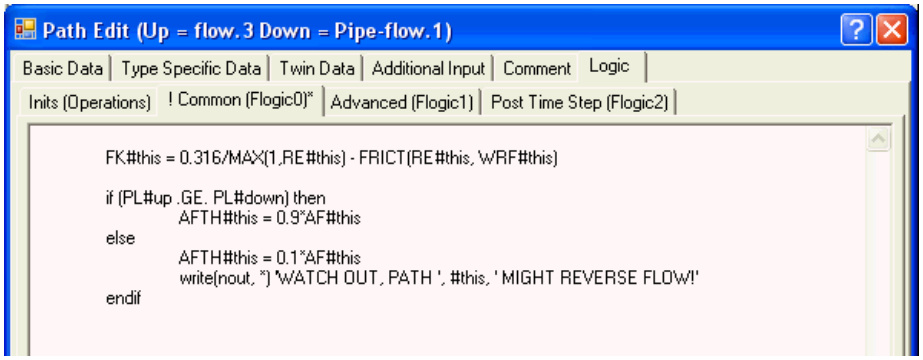


FIGURE 5-55. Example of Fluid Network Element Logic (for a Path)



As can be seen in these figures, four locations are available for placing network element logic, with each tab providing access to a different logic block such as Operations or Flogic 0 (as identified in Section 5.6.5 and Section 5.6.6). Each location corresponds to a different point within the SINDA/FLUINT solution at which the logic is invoked.* All of the locations can be used simultaneously for each element, but unless otherwise directed by the SINDA/FLUINT manual, "Common (Flogic0)" is the best location for most fluid customizations, and "Iteration Dep (Varbl1)" is the best location for most thermal customizations. For both thermal and fluid elements, the "Inits (Operations)" is the best location for any one-time initializations before a solution.†.

Note: The fluid FLOGIC0 tab and the thermal Iteration Dep tab have a "!" before the name to identify them as most common location for user logic. Once network based logic has been defined a logic tab, an "*" will appear after the tab name to indicate content on the tab.

* C-like expressions placed in data fields are updated continuously, whereas Fortran-like logic is executed at discrete and predefined points during a steady state or transient solution.

When network element logic has been employed, the generated logic, with the exception of initializations (Operations), will be placed *after* any instructions that have been added directly to a logic block using the Case Manager's Advanced tab. For example, the logic generated from the inputs in Figure 5-55 will be placed after any inputs the user might have placed in the FLOGIC 0 block of submodel "airflow." Therefore, network element logic will be executed after instructions that were placed directly in the logic block. The order network element logic from several network elements will be written out to the SINDA/FLUINT input file is not controllable. If logic is defined on two network elements and one must be executed before the other during execution, then the logic for both elements should be defined in the submodel level logic block rather than in network logic.

Any initialization logic defined in the "Inits(Operations)" tab will be placed in the subroutines S2TINITS (for thermal logic) and S2FINITS (for fluid logic). These routines are called internally by SINDA/FLUINT immediately after the appropriate build statement (BUILD or BUILDF). If a custom operations block is invoked by the user with multiple build statements, note that all logic defined in "Inits(Operations) will be called after each build statement.

Network element logic is extremely powerful because it combines the best features of data field (register-containing) expressions and concurrently executed user-defined Fortran logic blocks, and avoiding the major pitfalls of each:

1. A fixed list of built-in functions is available in data expressions. In logic blocks (including those generated using network element logic), *any SINDA/FLUINT support routines can be invoked, as can user-supplied subroutines and dynamic link libraries (DLLs).*
2. Data expressions are limited to a single line. In logic blocks (including those generated using network element logic), *any number of lines can be used* (though the width limit of 132 characters applies per line).
3. Some programming operators such as DO loops are not supported in expressions, but *any valid Fortran programming tool (including input/output) is available* in logic.
4. *User arrays and interpolations of arrays are permitted in logic*, but not in data field expressions.
5. *Logic can address and adjust two or more parameters simultaneously* (e.g., UB and UEDT for an HTU tie, or FC and FPOW for a tube or STUBE connector), whereas data expressions intrinsically result in the return of a single value.
6. Logic blocks are stored separately from network objects, and can become disconnected from them. For example, logic might refer to tank #1040 (e.g. "PL1040" in FLOGIC 2), and might cause an error if tank #1040 is renumbered or deleted, but the corresponding logic was not independently adjusted. *Data expressions, like network element logic, are stored with the object itself and so are deleted along with it and (using indirect operators) are insensitive to numbering changes.*

Some operations, such as checking for the termination of a transient run in either VARIABLES 2 or FLOGIC 2, should *not* be performed continuously as would be the case for a data expression (defining TIMEND in this case). Also, some parameters should be adjusted at some points in the solution (e.g. TLEN for a path in FLOGIC 0 versus FC/FPOW adjustments/inspections for a tube in FLOGIC 1). Logic blocks are invoked at predetermined points in the solution. This provides more control, but it does require the user to intelligently decide into which logic block an operation need be placed, whereas expressions are updated continually and therefore relieve the user of that responsibility (along with the control).

† Logic placed in these locations is inserted after the BUILD (for thermal) and BUILDF (for fluid) commands, and is re-executed if those build commands occur more than once inside an OPERATIONS block. Note that the user logic inserted in these "Inits" blocks will not appear in OPERATIONS itself, but rather inside SUBROUTINES DATA as special routines that are called within SINDA/FLUINT's BUILD and BUILDF routines.

5.6.8.1 Indirect Referencing in Network Logic

Inspection of Figure 5-54 and Figure 5-55 illustrates the key feature of network element logic: indirect operators such as "#this." Normally, such references are restricted to expressions in data fields, and are illegal in logic blocks because the SINDA/FLUINT preprocessor has no means of understanding to which node or lump an indirect operator refers. However, it is a feature of Sinaps that network element logic (and *only* network element logic) can use indirect references and all such references are converted into logic that the SINDA/FLUINT preprocessor can understand: indirect references such as "#this" and "#up" are converted into node or lump or conductor (etc.) IDs such as "42901" and "123" when the model is run.

This feature means that the user need not worry about element IDs, ***and it also means that one block of network element logic can apply to multiple network elements simultaneously.*** This is very useful when copying and pasting portions of a network containing network logic or editing multiple objects.

For example, the network element logic in Figure 5-54 initializes the capacitance of the current node as a function (specifically, a user-supplied function placed in SUBROUTINES and named MY_FUNCTION*) of the current node's temperature (T#this) and a register (EACHVOL). If many nodes were to be initialized using the same logic, all of them could be edited at once without regard to their naming schemes (i.e., IDs). In this respect, Sinaps network element logic can be viewed as a "logic generator."

Table 11below lists all the indirect operators available in Sinaps for use in Network Logic (note that submodel referencing is not available for use in data fields). When converting network logic in a submodel logic block, Sinaps will write out only the ID of the network element with no submodel prefix. This is useful for passing just the element ID or an element parameter (TL15) into a subroutine call:

```
CALL HLDLMP('#sub',#this)
```

Sinaps also recognizes if the object being referenced through indirect referencing is in a different submodel, for example a reference to the node ID or a parameter associated with the node in the network logic associated with a fluid tie will always be written out with the thermal submodel prefix.

If an indirect reference is being placed to a global logic block (e.g. OPERATIONS or PROCEDURE), the submodel prefix will always be included, unless the reference is just "#this" with no parameter reference prefix. For example, TL#this in the network logic associated with a lump flow.15 will be written to operations as flow.TL15.

The #num indirect reference allows the ID of a network element to be written out with no submodel prefix regardless of which logic block it is used in. This is useful when a series of numbered registers have been created to be associated with as series of elements, perhaps the registers POWER1, POWER2, and POWER3 have been created to represent the power to the crankshaft from cylinders (lumps) 1, 2, and 3 in an internal combustion model.The following network logic can be added to each cylinder lump to calculate the register values. By using the #num, the submodel prefix is suppressed regardless of which logic block the logic is assigned to.

```
POWER#num = (PL#this-PL1000)*VDOT#this
```

* Actually, this name exceeds the SINDA/FLUINT limit of 32 characters, which might cause conflicts if there is more than one function starting with the truncated "MY_FUNCTI." Furthermore, subroutines are preferable to functions because function names must avoid the spell checker in SINDA/FLUINT. The example in Figure 5-54 violates both of these guidelines: it is intended to convey potential modeling power and not to be used literally or as a template.

TABLE 11. Indirect Operators for Referencing Processor Variables in Network Logic

Element	Operator	Meaning
Node	#this	the identifier of the current node
	#num	the identifier of the current node, suppresses submodel prefix
	#sub	the identifier of the current submodel*
Conductor	#this	the identifier of the current conductor
	#num	the identifier of the current conductor, suppresses submodel prefix
	#sub	the identifier of the current submodel*
	#nodeA	the identifier of the first node
	#nodeB	the identifier of the second node
	#sub_nodea	the identifier of the submodel containing the first node*
	#sub_nodeb	the identifier of the submodel containing the second node*
Lump	#this	the identifier of the current lump
	#num	the identifier of the current lump, suppresses submodel prefix
	#twin	the identifier of the twinned tank (if applicable)
	#sub	the identifier of the current submodel*
Path	#this	the identifier of the current path
	#num	the identifier of the current path, suppresses submodel prefix
	#twin	the identifier of the twinned path (if applicable)
	#sub	the identifier of the current submodel*
	#up	the identifier of the upstream lump
	#down	the identifier of the downstream lump
	#uptwin	the identifier of the upstream lump twin
	#downtwin	the identifier of the downstream lump twin
Tie	#this	the identifier of the current tie
	#num	the identifier of the current tie, suppresses submodel prefix
	#twin	the identifier of the twinned tie (if applicable)
	#path	the identifier of the associated path (not valid for HTU, HTUS)
	#path2	the identifier of the second associated path (only valid for HTNC)
	#pathtwin	the identifier of the associated path twin (not valid for HTU, HTUS)
	#path2twin	the identifier of second associated path twin (only valid for HTNC)
	#lump	the identifier of the attached lump (not valid for HTUS, HTNS)
	#lumptwin	the identifier of the attached lump twin (not valid for HTUS, HTNS)
	#node	the identifier of the attached node
	#sub	the identifier of the current submodel*
	#sub_node	the identified of the submodel containing the attached node*
Ftie	#this	the identifier of the current ftie
	#num	the identifier of the current ftie, suppresses submodel prefix

TABLE 11. Indirect Operators for Referencing Processor Variables in Network Logic

Element	Operator	Meaning
	#twin	the identifier of the twinned ftie (if applicable)
	#lumpC	the identifier of the first lump
	#lumpD	the identifier of the second lump
	#lumpCtwin	the identifier of the first lump twin
	#lumpDtwin	the identifier of the second lump twin
	#path	the identifier of the associated path (for AXIAL fties only)
	#pathtwin	the identifier of the associated path twin (for AXIAL fties only)
	#sub	the identifier of the current submodel*
Iface	#this	the identifier of the current iface
	#num	the identifier of the current iface, suppresses submodel prefix
	#tankA	the identifier of tank A (reference tank)
	#tankB	the identifier of tank B (second tank)
	#sub	the identifier of the current submodel*

* Submodel referencing is only applicable to network logic and cannot be used in data fields.

5.6.8.2 Network Logic Include Files

To support large blocks of network element logic, Sinaps allows the use of include files which use #this referencing. To define an include file use the format #include [filename] placed in column 1 as shown in the example below. The #include format is only allowed in network element logic.

```
#include myobjectlogic.inc
```

When the #include statement is found, Sinaps will translate all #this indirect referencing in the include file and insert the converted logic into the appropriate location in the *.sin file. Unlike traditional include files, Sinaps does allow the nesting #include files.

For more in-depth information process variables and indirect referencing, please see “Processor Variables” on page 6-8

5.7 Thermophysical Property Manager

The thermophysical property database file contains physical property values for solid materials (specific heat, conductivity, and density) catalogued under a user defined name. By default an empty property file, TdThermo.tdp, is created in the working directory when a model is first created. The user has the option of opening/importing an existing file (Open button at the top of the form, as shown in Figure 5-56), or creating a new property file (Create button). When a property file is opened, an association to the file is automatically created between the Sinaps model file and the property file. This option is useful when several engineers are using a common set of property data or when sharing a property descriptions with someone using Thermal Desktop® (the *.tdp files are formatted the same).

Once a property file is associated with the model, it can be edited through the *Thermophysical Properties* window, depicted in Figure 5-56 and accessible through the pull down menu View>Thermophysical Properties. If the default new property file is being used, the user must specify the units for the thermophysical properties file. Radio buttons allow the user to specify the units as SI or ENG units as defined in Table 12. When adding a material to the database, the originally selected unit convention must be followed. The units of the thermophysical property file do *not* need to be the same units as the model. Sinaps will convert the units as necessary when applying the properties.

> Caution: When a material is deleted from the thermophysical property database, other models which share the database will fail to run properly if they are still referencing the now deleted material. The error displayed when a model is run states "The material {name} wasn't found in the database - the output will be invalid."
>
> The material pull down menu in the node and conductor edit forms are based on properties which exist in the database, so these forms will not display missing properties. A "broken" model will only display the missing property in the Object Properties panel when editing an object or when the network is colored by material. When coloring by material, the object with a missing material description will display gray.

FIGURE 5-56. Thermophysical Properties Window

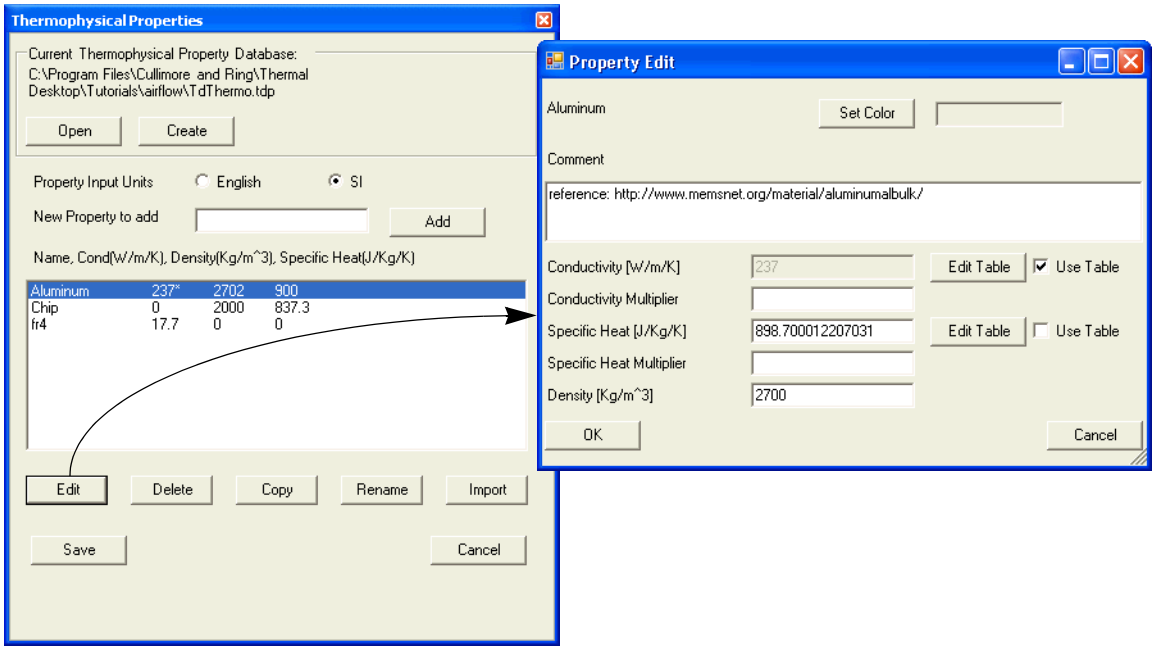


TABLE 12. Thermophysical Property Units

	English	SI
Conductivity	BTU/hr-ft-R or BTU/hr-ft-F	W/m-K or W/m-C

TABLE 12. Thermophysical Property Units

	English	SI
Specific heat	BTU/lb_m-R or BTU/lb_m-F	J/kg-K or J/kg-C
Density	lb_m/ft^3	kg/m^3

Creating a new material property: Simply type the name by which the material will be referred (e.g, "steel" or "AL6061") in the New Property to add field, then click the Add button. This action will bring up the Property Edit form shown on the right side of Figure 5-56. The user has the option of providing a comment to help with model documentation, possibly listing the reference for the source of the material properties. The user can also set a unique color for each material. This color will be used to color the network by material, an input check option (see "Color by Material" on page 4-18).

When inputting property data, the user must provide a value or expression in the each of the property fields. Unfortunately, *thermophysical properties cannot be defined using registers* in order to retain compatibility with Thermal Desktop. To allow parametric studies to be performed on thermophysical properties, a "multiplier" field has been provided for each property where the user *can* define a register to be used as a multiplier. (Any registers used in these multiplier expressions must be present in any models that reuse the same property database.)

For example, a register named KFACT could be created and defined as the conductivity multiplier. This would enable a parametric to be run from the Case Manager in which KFACT varies from 0.8 to 1.2, perhaps as needed to test the uncertainty this property.

Sinaps does not support anisotropic material properties since, unlike Thermal Desktop, it does not contain a description of geometric objects.

Temperature-dependent properties: By clicking the Edit Table button, temperature-dependent property data can be defined. This data is expected to be comprised of a column of temperature-value pairs, as shown in the left pane of Figure 5-57. The input window provides a graph of the data as a visual verification of the inputs.

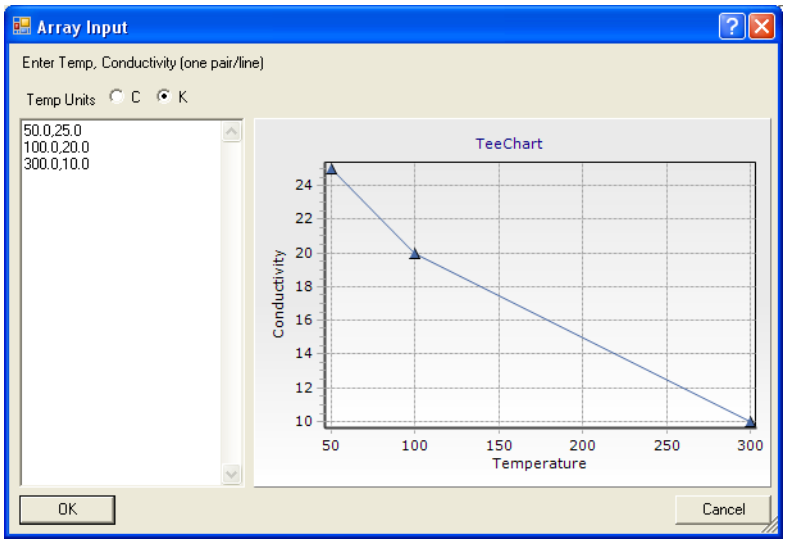
When inputting the data, the temperature can be input using absolute or standard scales for the current units (English or SI). The Temp Units radio button allows the user to select the temperature units which will be input prior to creating a table. Once input and the form is exited, Sinaps will convert temperatures to absolute. If the table needs to be modified at a later point in time, the radio buttons can be used to toggle the table back to standard temperature.

Note that the temperature portion of the temperature-value pairs must be monotonically increasing throughout the table. If not input correctly, upon exiting the Array Input dialog Sinaps will rearrange the pairs to ensure increasing temperature.

To invoke the use of the temperature-dependent data, the user must select the Use Table check box.* Note that in the example shown in Figure 5-57, temperature-dependent data is defined for the thermal conductivity. Therefore the value in the data field for thermal conductivity in Figure 5-56 is shown in gray to visually alert the user that temperature-dependent data is being used. The main property window displays an asterisk ("*") next to the value of the thermal conductivity for aluminum, again alerting the user that temperature-dependent properties are being used.

* This two-step process provides a mechanism for entering a table of variable values, and then choosing to either use that table or a constant value (without deleting the underlying table).

FIGURE 5-57. **Temperature Dependent Property Inputs**



Editing/Deleting an existing material property: Inside the thermophysical property manager, select the material to be edited from the list of available materials and then click the Edit or Delete button.

> Caution: This delete action cannot be reversed with a Ctrl-z. If it is performed accidently, press Cancel, or exit the form without saving.

Copying a material property: The user may copy and paste materials to create a new material. Simply highlight the material to be copied and select the Copy button. A prompt for a new (and unique) property name will then appear.

Renaming a material property: The user may rename a material by highlighting the material and then clicking the Rename button. After providing a new name a prompt will appear: "Save material database and rename material in model." Selecting "no" causes the material name in the thermophysical property database to be changed, but the new name will *not* be propagated throughout the model (i.e., node and conductor definitions). To instead propagate the name change throughout the model, select "yes."

Importing thermophysical property databases: Once a thermophysical property file has been created, other Sinaps or Thermal Desktop property databases can be imported and merged with it. Click the Import button to start the import process. A browse window will appear. Navigate to the property file to be imported. Upon importing, the content of the file will be merged with the existing property file in the model. *If duplicate property names are found, the property from the file being imported will take precedence.*

> Caution: This delete action cannot be reversed with a Ctrl-z. If it is performed accidently, press Cancel, or exit the form without saving.

Sinaps does not support some of the Thermal Desktop property features such as anisotropic materials, pressure dependent properties, effective emissivities, ablation or fusion. When importing such materials, some data will not be available when viewed in Sinaps although it will remain in the property file for continued use with Thermal Desktop:

1. For **anisotropic materials**, only the x-direction properties will be imported.
2. **Pressure dependent** conductivity will be ignored resulting in the import only of the value in the conductivity field (possibly grayed-out in Thermal Desktop.)
3. For materials defined using Thermal Desktop **symbols or expressions**, the symbols and expressions will not be imported, but the *value* of the expression *will* be imported.
4. **Effective emissivity** is ignored and not imported.
5. **Albative** properties are not imported.
6. **Fusion** properties are ignored.

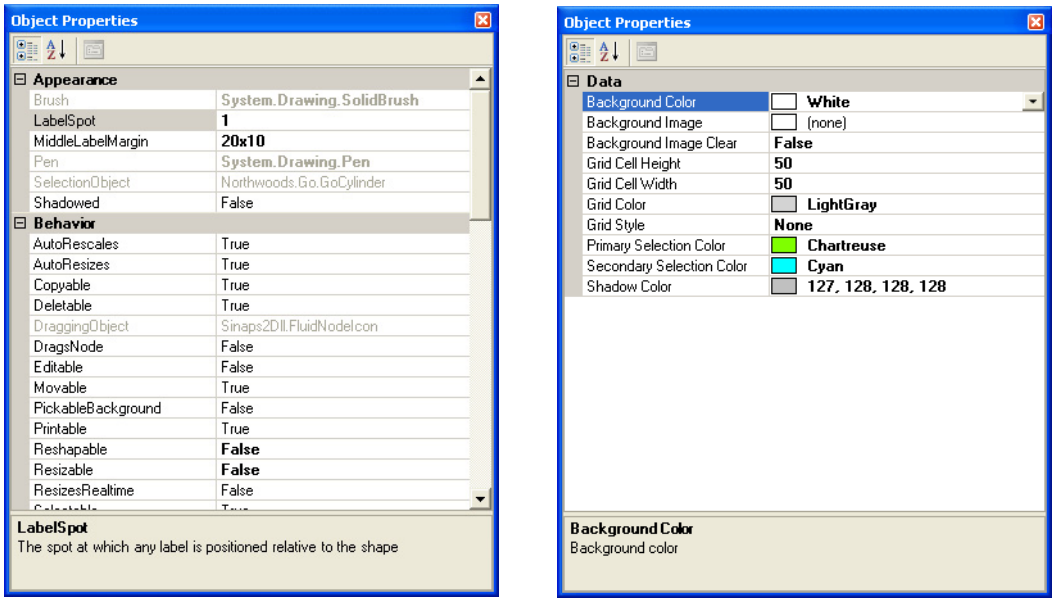
5.8 Object Properties

The Object Properties dialog is a task panel which allows the user to edit the underlying data for all objects in Sinaps including network elements (their input parameters and visual representation on the network diagram), the visual appearance of the network workspace, network graphical objects, and more. If the Object Properties panel is not open, it can be opened using the main Sinaps menu `View>Properties`.

The Object Properties panel can be used instead of the normal edit window for most editing operations in Sinaps. Some items can *only* have their properties edited using the Object Properties panel, such as network appearance (e.g., background art). Examples of a fluid icon properties listing (left) and a network properties (right) are shown in Figure 5-58.

The properties window is in the format of a list of properties in the left column, with their values or setting listed in the right column. In some cases the properties are categorized based on data, appearance, or behavior. Categories can be expanded or collapsed. Alternatively, properties can be listed alphabetically (by pressing), returning the to categorization as needed (by pressing). Properties which are gray cannot be changed by the user.

FIGURE 5-58. Object Properties Task Panel



5.8.1 Editing Network Properties

The Network Properties dialog (right side of Figure 5-58) allows the user to customize the look of the network workspace. The user can select a background color. Note that black is not recommended since the conductors will not be visible against it.

A grid is provided for the network layout. By default the grid is not visible. The grid can be toggled on or off using the grid toggle icon ( ) or using the network Layout menu. The appearance of the grid can be changed to display as dots, crosses, lines. The color of the grid may also be changed.

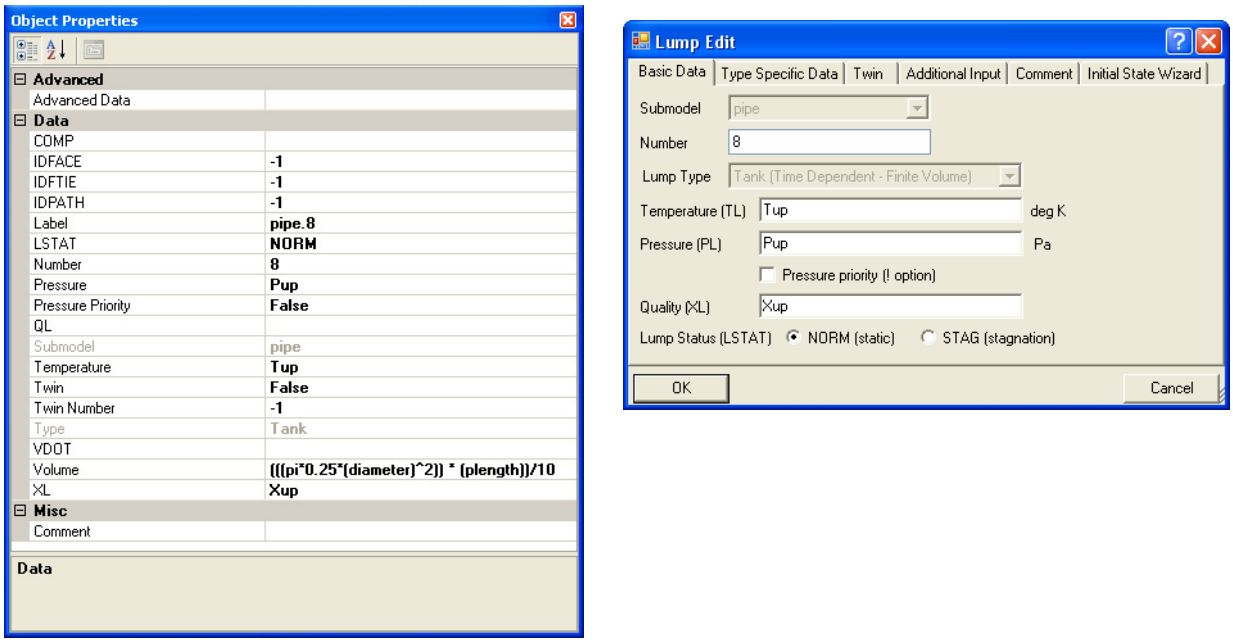
The user can change the colors used to denote object selection. When a network object is selected, a colored box will appear around the object. Selected network links will appear with colored grip points at each end of the link. The selection color controls the color of the selection box and the grip points.

5.8.2 Editing Network Element Input Parameters

The Object Properties dialog can be used for editing network input parameters. The content of the dialog will vary depending on the type of object selected. Figure 5-59 shows edit information for a typical lump using both the Object Properties dialog and the edit form. In general all parameters are available through both the edit form and Object Properties. However, there are a few fields which can *only* be changed through the Object Properties such as the label for a network object. Note that in many cases the input descriptors in the Object Properties are parameter names, not descriptions. In most cases users will find the edit forms easier to work with. However, advanced users who know the parameter names may find the Object Properties dialog to be a time saver.

Note: Unlike the network element edit forms, the object properties dialog does not provide built-in error checking. However other input checking methods are available, see "Input Checking" on page 4-17.

FIGURE 5-59. Network Element Input Parameters



5.8.3 Editing the Visual Appearance of Network Objects and Links

The user can modify the visual appearance of object icons and links using the Object Properties. This feature is accessible by selecting an object or link and then using the right click object edit menu. The resulting properties list, although extensive, is currently somewhat limited. This feature is mostly available to support future expansions of Sinaps.

5.8.4 Editing Network Graphical Objects

Network graphics such as lines, circles and squares can be edited using the Object Properties dialog. After creating a graphical object, click on the object to select it. Once selected, its properties will appear in the Object Properties dialog. Circles, rectangles, and lines may have their color and line width adjusted. Lines can also have arrow locations and arrow styles modified.

5.8.5 Editing Post-processing Objects

Post-processing gauges and meters can be modified using the Object Properties dialog. Content of the gauge or chart can be changed through the object edit window accessible by double clicking on the object while the appearance of the

object is controlled through the Object Properties Dialog. Parameters which can be changed include (but are not limited to) scale tick units and tick frequency, scale maximum and minimum values, line thickness, and more. For more information on meters and gauges, see “Post-processing Tools: Meters, Gauges and Thermometers” on page 10-18.

5.8.6 Editing Post-processing Color Scales

The post-processing color and thickness scales can be customized using the Objects Properties dialog. Click on a scale to select it and the properties will appear in the Object Properties panel. The user can modify autoscaling, minimum and maximum values, number of division, number format, and absolute value. See “Customizing Post-processing Scales” on page 10-17 for more information.

5.9 Using INCLUDE Files

The user can specify external files to be included in the model file. These included files may contain fluid property information, test data, user logic, user-defined subroutines or network element definitions. When a file is included it is recommended that the INSERT statement be used rather than the INCLUDE statement since INSERT allows nesting. Working fluid include files are discussed in “Working Fluids” on page 5-19. The user can elect to insert a external logic file by placing an INSERT statement in any logic or subroutine block. INSERT statements must be placed in column 1 as shown below. If the file is outside the working directory, either the path should be supplied in the statement or defined in “Include Paths” on page 9-17:

```
INSERT f9102.inc

INSERT C:\MYPATH\f9102.inc
```

Network Includes: An INCLUDE file may be used to define portions of the network (for example, some nodes and conductors that do not need to be depicted graphically, perhaps because they are too numerous or were created by other tools). Elements defined in INCLUDE files will not appear in the network, however they will be part of the solution. These INCLUDE files need to be specified using the Case Manager, see “Network Includes” on page 9-30.

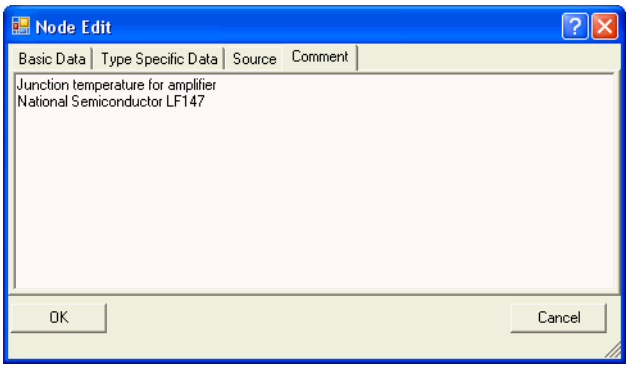
5.10 Documenting Models

This section describes the tools for model documentation. It is important to provide sufficient documentation when creating a model. Thorough documentation allows a new user to quickly understand a model when inherited from a co-worker. It will also assist a customer who may receive a model as part of a deliverable for a contract. Sinaps allows the user to fully document the model directly in the model file thus eliminating additional files and paperwork.

Network Comments: The user can place comments directly on the network. To add a network comment click on the comment icon (C) or drag the comment icon from the Tool Pallet to the network field. After placing the comment, double-click on it to edit the contents. The user may use these comments to identify portions of the network, flag issues, or provide documentation regarding network modeling methods and decisions.

Element Comments: Every network object, link, or macro has a Comment tab associated with it allow the user to provide any necessary documentation regarding the object. Such comments may include component part numbers, property references, documentation of heat loads, or decisions used for selection of element types.

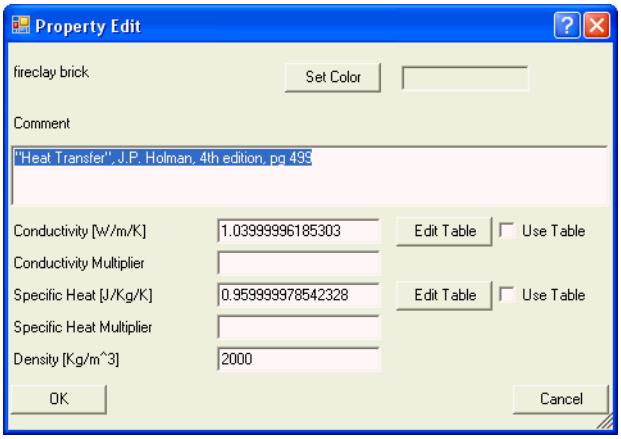
FIGURE 5-60. Example of an Element Comment



Case Manager Comments: For each case defined in the Case Manager, the lower portion of the Case Sets tab contains a comment field. Simply click in the field to add a comment. The user can add descriptions or comments in this area pertaining which are case-specific.

Thermophysical Properties: When adding a new material to the thermophysical properties dialog, a comment may be added as shown in Figure 5-61. Such a comment may provide reference to the literature or testing performed for the thermophysical properties.

FIGURE 5-61. Example of a Thermophysical Properties Comment



CHAPTER 6 Parameterizing a Model

Sinaps provides the ability to parameterize a model. A parameterized model allows for the rapid and complete propagation of changes, provides a convenient means for model documentation, enables easy-to-run parametric studies, and provides access to advanced options such as goal seeking, sizing/optimization, model correlation, and reliability engineering methods. The key to model parameterization is using *registers* and register-based *expressions* throughout a model.

Registers are variables that are arbitrarily created and named by the user for use in data fields, expressions, and logic throughout the model. Registers are *global,* which means they are not specific to any one submodel: registers can be used in expressions throughout the model. They can also be defined as expressions containing other registers or translatable SINDA/FLUINT processor variables such as "sub.T22," "timen," "left.drlxcc," or "pipe.FR20." Registers can be defined as either single or double precision or as an integer.

Processor variables are parameters whose value is not known until SINDA/FLUINT execution begins, and which may change during processor execution. They require special treatment because their value is unknown to Sinaps expressions, except during post-processing.

With few exceptions,* anywhere a value is input into a data field or logic block, an expression can be supplied instead. ***Expressions*** can use multiplications ("*"), divisions ("/"), additions ("+"), subtractions ("-"), and exponentiations ("^" or "**") nested within arbitrary levels of parentheses. Furthermore, the user can use built-in functions (like sine, cosine, and logarithms), built-in conversion constants and physical constants (pi, and the Stefan-Boltzmann constant). Expressions can also contain conditional operations (equivalent to IF/THEN/ELSE in Fortran) and may reference any processor variable. Examples are provided below, with more background available in Section 2.8 of the SINDA/FLUITN User's Manual.

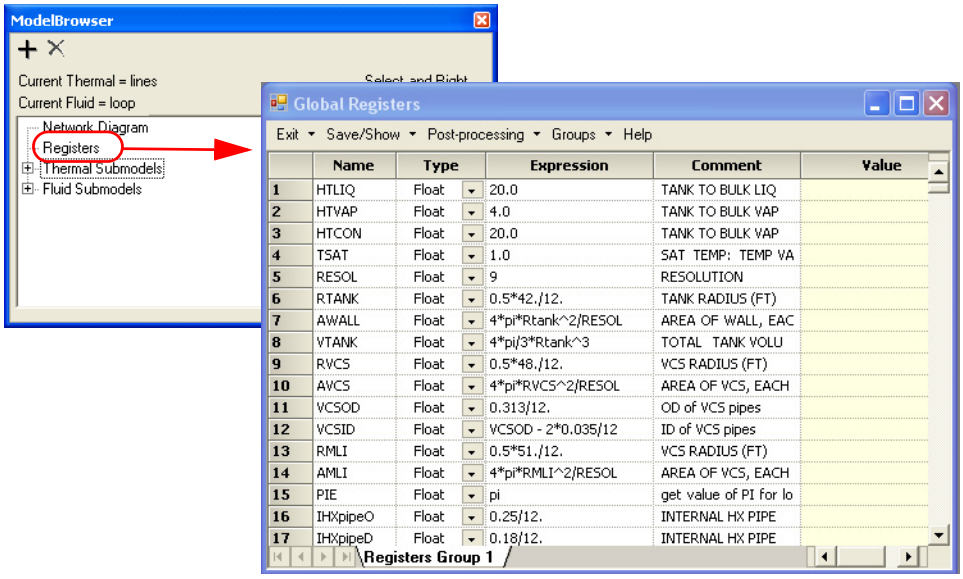
* As an example of an exception, an element ID (e.g., a node number) cannot be defined as a variable; it must be specified as a simple integer.

6.1 Defining Registers

To access registers in a model, open the Global Registers table or "spreadsheet" using the main menu option View>Registers (alternatively registers can be access through the model tree in the Model Browser task panel as shown in Figure 6-1). The register table is referred to as spreadsheet since it has a built-in calculator function displaying the value for each register. These values are displayed when the spreadsheet is saved using the pull-down menu Save/Show>Save & Show Values. Registers based on undetermined processor variables will be initialized to a random number since the value of processor variable is unknown until SINDA/FLUINT begins execution. For case specific registers, see "Case-specific Registers" on page 6-7.

In the register table columns and rows can be resized. To change the width of a column, select the column breaks in the column header line and drag the break horizontally to its new dimension. Likewise, to change the height of a row, select the break line between row numbers on the left and raise/lower the break to the new location. Using these resize options, the size of a cells can be change to show the entire contents. Note that only the column width customizations will be saved with the model file.

FIGURE 6-1. Accessing the Register Table



	Name	Type	Expression	Comment	Value
1	HTLIQ	Float	20.0	TANK TO BULK LIQ	
2	HTVAP	Float	4.0	TANK TO BULK VAP	
3	HTCON	Float	20.0	TANK TO BULK VAP	
4	TSAT	Float	1.0	SAT TEMP: TEMP VA	
5	RESOL	Float	9	RESOLUTION	
6	RTANK	Float	0.5*42./12.	TANK RADIUS (FT)	
7	AWALL	Float	4*pi*Rtank^2/RESOL	AREA OF WALL, EAC	
8	VTANK	Float	4*pi/3*Rtank^3	TOTAL TANK VOLU	
9	RVCS	Float	0.5*48./12.	VCS RADIUS (FT)	
10	AVCS	Float	4*pi*RVCS^2/RESOL	AREA OF VCS, EACH	
11	VCSOD	Float	0.313/12.	OD of VCS pipes	
12	VCSID	Float	VCSOD - 2*0.035/12	ID of VCS pipes	
13	RMLI	Float	0.5*51./12.	VCS RADIUS (FT)	
14	AMLI	Float	4*pi*RMLI^2/RESOL	AREA OF VCS, EACH	
15	PIE	Float	pi	get value of PI for lo	
16	IHXpipeO	Float	0.25/12.	INTERNAL HX PIPE	
17	IHXpipeD	Float	0.18/12.	INTERNAL HX PIPE	

Creating a register: To create a new register simply type the information on a line of the register spreadsheet. Each registers will require the following inputs:

- *Name* - An alphanumeric string by which the register will be referred in expressions and data fields. Register names must start with a letter (and not a number), and they cannot exceed 32 characters in length, and (like the rest of SINDA/FLUINT inputs) are case insensitive. Their names cannot conflict with any reserved word (see Table 6-1 in the SINDA/FLUINT User's Manual). Underscores are allowed in registers names however other special characters (%, ^, &, #, etc.) are illegal.
- *Type* - A flag designating if the registers is to be treated as a real number ("Float" option) or an integer ("Int") in logic. Note that a third option ("Double") is also available. This option can be used to indicate a double precision

register in versions *prior* to 5.4. Starting with version 5.4, by default registers are double precision, making the "Double" and "Float" options identical.

To flag a register as an integer, set the Type to "Int." *This option is only needed if the register needs to be an integer in Fortran-based logic blocks,* either as an argument in a subroutine expecting an integer, or as needed to facilitate comparisons (e.g., "IF(MyReg .eq. 1) ..."). If the value defined for the register is a real number (e.g., 3.4) but the Type is set to "Int," the value of the expression or number will be rounded off (e.g., 3.0). Rarely does a variable need to be an integer. Exceptions include variables that are used to control logic (via IF-type checks in logic blocks), or that contain fluid identifiers to be passed to the processor for use in a CHGFLD call, or that contain node or array numbers for use in logic block operations, etc.

- *Expression* - A value or mathematical formula that defines the register's value. This might simply be a number, but it might also be an expression referencing other registers or processor variables. Possible mathematical operations are defined below.
- *Comment* - Each register has a comment field so that a detailed description about be entered. These comment fields allow other users to more easily understand a given (inherited) model, and might even contain usage instructions for a model.

Editing a Register: A register can be edited directly in the spreadsheet. For long expressions or comments, adjust the columns widths and rows heights to make the entire cell visible.

Renaming a Register: Renaming a register requires more than simply changing the name of the register in the register dialog. Once a model has been built and the register used, all occurrences of the register in expressions and logic must also be changed. The best way to rename a register is to use the Find and Replace tool which will search and replace all occurrences of the register throughout the entire model. For usage information on find and replace, see Section 3.4.1.4.

Deleting a Register: To delete a register, select the associated line in the spreadsheet and either hit the delete key or right click to bring up the pop-up menu and select Cut. The now-empty row will disappear the next time the form is opened.

> Caution: This operation cannot be undone using Ctrl-z, so if its use is accidental, exit without saving the form using Exit>Quit Window, or use Save/Show>Restore old values to return to a previously saved state.

Saving Registers: Under the Save/Show pull-down menu are options for storing and retrieving registers and their associated comments. Save/Show>Save Values writes the current contents of the register window to memory (not to disk!). This is a *local* save. As with other local saves, the menu option File>Save is required to write the contents of memory to the hard drive. If the user wishes to undo changes made to registers that have not been saved, the Save/Show>Restore old values option can be used to return the window's state to the last time it had been saved.

Sorting Registers: It is often easier to find registers in a long list if they are sorted alphabetically. Registers can be sorted alphabetically using the pull-down menu Group>Sort.

Show Values: The Save/Show>Save & Show Values can be used fill in the "Value" column with current calculated values. If a register is defined based on a processor variable, the value will be set to a random value since the processor variables are undefined outside of a SINDA/FLUINT run.

> Note: The value column can also be used for post-processing to show the current values at a specified time step or record time from a save file. See "Post-processing Operations" on page 10-9 for details on this and other register post-processing options.

Built-in Registers. For convenience, various physical parameters and common unit conversions are predefined and available for use in any expression. This list of built-in constants is accessible from the register table using the pull-down menu Save/Show>Show Builtin Values. If the user creates a register of the same name as a built-in parameter, the defined register value will override the internal value for that model, but no warning will be issued.

By default built-in constants are written to the .sin file as registers, allowing these built-in constants to be used in logic blocks. The user can suppress the writing of constants as registers by toggling the register dialog menu option Save/Show> Set Builtins as Registers. Note that when turned off, these constants are no longer available for use in logic blocks.

Built-in Functions. Common mathematical functions are also available for use anywhere within an expression, as shown in Table 1. Their arguments (such as "float" in the following table) can be values, register names, or subexpressions.

> Note: These functions, which are only available *within expressions*, should not be confused with similar Fortran functions used in user logic, since those functions may operate somewhat differently. For example, "max" can only have two arguments in these expressions, whereas any number of arguments may be used in the Fortran intrinsic function of the same name.

TABLE 1. Built-in Functions

Function	Description	Example
abs	absolute value	abs(float)
acos	arccosine (radians)	acos(float)
asin	arcsine (radians)	asin(float)
atan	arctangent (radians)	atan(float)
ceil	round-up (to next larger integer)	ceil(float)
cos	cosine (given radians)	cos(radians)
exp	base-e exponentiation (opposite of "ln")	exp(float)
floor	round-down (to next smaller integer)	floor(float)
ln	natural (base-e) logarithm	ln(float)
log	base-10 logarithm	log(float)
max	return larger of two values	max(float1,float2)
min	return smaller of two values	min(float1,float2)
sin	sine (given radians)	sin(radians)

TABLE 1. Built-in Functions

Function	Description	Example
sqrt	square root	sqrt(float)
tan	tangent (given radians)	tan(radians)

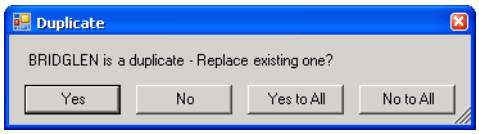
6.1.1 Importing/Exporting Registers

Registers can be exported to a file ("registers.xml" by default) for sharing with other Sinaps models or other Sinaps users using the Save/Show>Export to File option. The export feature allows the export of all the registers (all groups), a single register group, or a user defined selection of registers within a single group.

Registers can be imported using the Save/Show>Import from File option. Currently Sinaps does not support import of a register file from Sinaps*Plus* (*.bin) or Thermal Desktop symbol file (*.sym).

When syncing different models that employ the same register names, it is common for the same register to exist in both the current model and the file being imported. When this is detected, a dialog such as is shown in Figure 6-2 will appear. Choosing Yes will overwrite the current definition of the conflicting register ("BRIGLEN" in the example shown) with the definition (expression, etc.) contained within the *.xml or *.sym file. Choosing No will leave the current definition in place, ignoring the definition in the file that is being imported. If there are many such conflicts, the user can choose to accept all imported redefinitions (Yes to All) or reject any imported definitions, preserving the definitions for registers currently shown in the register spreadsheet (No to All).

FIGURE 6-2. Resolving Conflicts when Importing Registers



6.1.2 Register Groups

Registers may be organized into groups displayed on separate worksheets at the discretion of the user. Each group is identified by a tab along the bottom of the register window. Additional register groups may be added using the pull-down menu Pages>Add Group. Each register group is limited to 1000 registers, with an maximum of 15000 registers permitted in any one model. All registers groups are collectively referred to as "global registers" and can be referenced by any submodel, hence a unique register name must be used, even if the registers are in different groups.

Moving Registers Between Groups: To move a register to a new group simply highlight the associated line in the spreadsheet by clicking on the row number (the first column) and use the pull-down menu Groups>Move to Group. Multiple registers may be selected by holding down the shift or control key while selecting rows.

Renaming a Register Group: When viewing a register group in the spreadsheet, the group can be renamed using the pull-down menu Group>Rename Group.

Deleting a Register Group: When viewing a register group in the spreadsheet, the group can be deleted using the pull-down menu Group>Delete Group. *This will delete all registers within the group.*

> Caution: This operation cannot be undone using Ctrl-z, so if its use is accidental, exit without saving the form using Exit>Quit Window, or use Save/Show>Restore old values to return to a previously saved state.

6.2 Using Registers and Expressions

Once defined, a register may be used as in any logic block or as part of an expression in any numeric data field. Expressions may be used any place a real (floating point) value is required within the SINDA input file data blocks: node temperatures, capacitances, SIV factors, tank volumes, tie conductances, tie path factors, etc. The final result of an expression is rounded to an integer if it is to be placed into an integer data field (e.g., the IPDC selection for tubes and STUBE connectors). ***Expressions*** can be defined as:

1. a floating point value (e.g., “1.0”, “3.”, “3.0e-7”);
2. a combination of floating point or integer values, perhaps using internal built-in functions (e.g., “-2.0*4”, “11-2*3”, “4.5*pi*sin(4.0)”) and conditional operations (e.g., “(a>b)?c:d”);
3. a register name, or combination of register names (e.g., “temp”, “len1+len2*cond”)
4. a reference to a processor variable (e.g., “t22”, “flow.q214”, “mysub.drlxcc”, or “fr#this”)
5. a combination of any or all of the above (e.g., “2.0*pi*cond*TLEN22/ln(router/rinner)”).

In general, expressions can be arbitrarily complex* and interrelated, with no dependence on input order. Such interrelations are resolved when the registers are saved or updated, and changes are propagated throughout the model at a various points during the network solution. Automatic updates occur after user logic blocks have been executed such that the network is “fresh” before a solution step occurs, but updates do not occur *during* the execution of any logic block. The user can elect to propagate changes *within* logic blocks by using the UPREG subroutine (please see section 7.11.1 of the SINDA/FLUINT User’s Manual for more information).

There are some restrictions to the expressions that may be used to define a register. First, a register cannot be defined in terms of itself (self-referential); it must be defined in terms of other values or registers. For example, one cannot define “a=a” or “a=ln(b/a).” Second, circular references must be resolvable. In other words, setting both “a=b” and “b=a” is inappropriate, whereas “a=b” and “b=c” and “c=pi” is acceptable.

6.2.1 Conditional Operators

It is often convenient to choose between one or more values within an expression. Conditional operators, equivalent to IF/THEN/ELSE in Fortran, may be used to accomplish such elections. Conditional operators can be used in any data field where an register or expression is allow, however they cannot be used in logic blocks (where Fortran rules apply

* Subject only to the 1000 character line length limit for data fields in a SINDA/FLUINT input file (132 characters for versions prior to 5.3).

instead). For example, simple thermostatic heater control logic could be implemented by defining the source on a node as:

(T22 > 25.0)? 0.0 : 50.0

meaning that if the temperature of node 22 exceeds 25 degrees, then turn the heater off (set power to zero), else turn it on to 50 units of power.

The format of an conditional expression is:

(expr1 OP expr2)? expr3 : expr4

where OP may be >, >=,<, <=, == (equal), or != (not equal). Subexpressions (e.g., "expr1") may themselves contain nested conditional operator expressions.

Another common use for a conditional statement is setting the problem end time when a transient simulation is to be run until a certain temperature condition occurs. In this case the user can define the problem end time (TIMEND) in the Case Manager to be the following:

(wall.T51 > 110)? timen : 1.0e6

This statement checks the temperature of node 51 in the submodel "wall" to see if it has exceeded 110 degrees. If that condition is met, the end time is set to the current time (global constant "timen") and the solution process is halted. If the condition is not met, the transient simulation continues using a large value (1.0e6 in this case) for the problem end time. Notice that this expression is defined at the global level (since timend is a global constant) and therefore the submodel name for the node 51 ("wall") must be included in the conditional statement. Otherwise, the submodel prefix is optional when working with the fields defining any particular network element, or other cases in which the submodel is known to Sinaps.

6.2.2 Case-specific Registers

Register values can be overridden for a particular case. Likewise, new registers can be defined which only apply to a specific case. Both options are invoked using the Registers option in the tree under the Advanced tab of the Case Manager. The case-specific register spreadsheet shown in Figure 6-3 will appear.

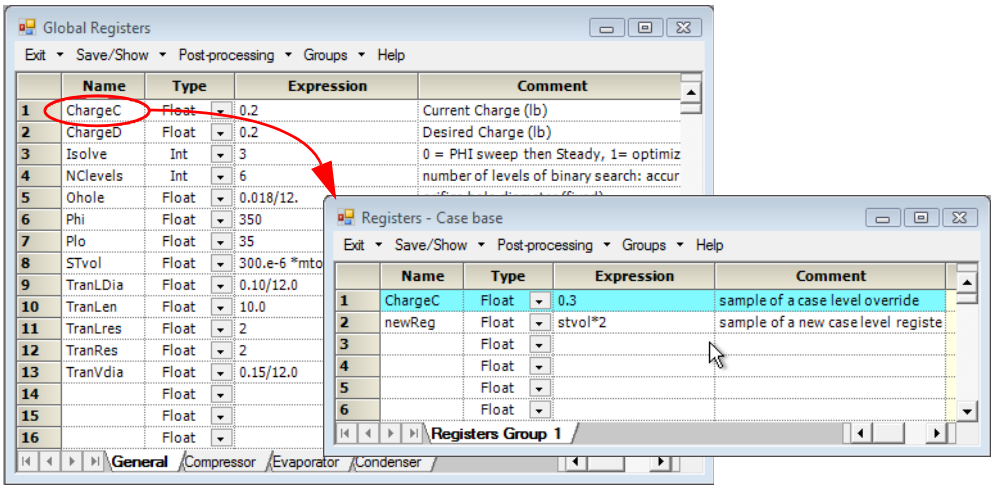
The new sheet that appears is named "Registers - Case *case_name*," where *case_name* is the name listed for the current case (default: "base"). Entering the name of a global register in the case-specific sheet overrides the global definition for this case only. A list of global registers can be readily accessed by placing the cursor in the name field and clicking the middle mouse button. Select the name of the global register and it will be placed in the name field. If a global register name is input in the case specific form, the row will automatically highlight in color to identify the input as a global override versus a case unique register. Adding a new name to the form creates a "local" or ***case-specific register*** that exists only when this case is executed.

Case-specific registers can be defined based on any global register or processor variable using the same rules described above for defining registers. Case-specific ("local") registers should *only* be used within the Case Manager for case-

specific control data, logic blocks, array data, etc. Otherwise, if a case-specific register is encountered in a different case (or in a global location), an error will result since that register will not exist in any other case.

In the example shown in Figure 6-3, the case-specific register window is titled “Registers - Case base” identifying it as registers specific to the case name “base”: compared to the “Global Registers” window displayed in the background. In that example, the register *ChargeC* is recognized automatically to be an override since a register with the same name exists in the global register list (it is highlighted in Figure 6-3 for clarity). The register *newReg* does not exist in the global register list, so it will be added for this “base” case only. In other words, *ChargeC* is being overridden, while *newReg* is being added as a case-specific register for the “base” case.

FIGURE 6-3. Case-specific Registers


Global Registers
Exit Save/Show Post-processing Groups Help
Name Type Expression Comment
1 ChargeC Float 0.2 Current Charge (lb)
2 ChargeD Float 0.2 Desired Charge (lb)
3 Isolve Int 3 0 = PHI sweep then Steady, 1= optimiz
4 NClevels Int 6 number of levels of binary search: accur
5 Ohole Float 0.018/12.
6 Phi Float 350
7 Plo Float 35
8 STvol Float 300.e-6 *mto
9 TranLDia Float 0.10/12.0
10 TranLen Float 10.0
11 TranLres Float 2
12 TranRes Float 2
13 TranVdia Float 0.15/12.0
14 Float
15 Float
16 Float
General Compressor Evaporator Condenser
Registers - Case base
Exit Save/Show Post-processing Groups Help
Name Type Expression Comment
1 ChargeC Float 0.3 sample of a case level override
2 newReg Float stvol*2 sample of a new case level registe
3 Float
4 Float
5 Float
6 Float
Registers Group 1

6.2.3 Processor Variables

The user can refer to processor variables within any expression. A ***processor variable*** is a reference such as “smn.T33,” meaning the temperature of node #33 in submodel “smn.” Processor variables are unique in that they have no value until the SINDA/FLUINT processor starts: they are assigned random values until the simulation begins. The use of processor variables in expressions represents an extremely powerful option that simplifies many modeling tasks. In essence, each data entry field can become a mini-logic block.

Almost any variable that can be referenced in logic blocks can be referenced within an input expression, including thermal and fluid network inputs (e.g., “G” for conductance, “DH” for hydraulic diameter), results (e.g., “T” for temperature, “FR” for flowrate, etc.), and status and control parameters (e.g., “LOOPCT,” “DRLXCC,” etc.).

Indirect Referencing: When defining an expression in any data field for a network element, that expression will often refer to attributes of that element itself, or of adjacent elements. For example, in the previous example defining the source on lump number 22:

(TL22>25.0)? 0.0 : ((TL22<20.0)? 50.0 : QL22)

the source was defined on the basis of the temperature of the lump (number 22) itself. This could equivalently (and preferentially) been input using the indirect operation "#this" as:

(TL#this>25.0)? 0.0 : ((TL#this<20.0)? 50.0 : QL#this)

Indirect referencing makes it easier to generate multiple elements (as an alternative to the increment operator described above), and to edit multiple elements or reuse the same expression in multiple locations. These operators are particularly convenient since the underlying expressions do not change when the model is resequenced or when the element identifiers change for whatever reason. See Table 2 for a full list of indirect referencing options for Sinaps data fields. For more in-depth information on the use and restrictions of process variables, please see Section 2.8.8 of the SINDA/FLUINT User's Manual.

TABLE 2. Indirect Referencing in Sinaps Data Fields

Element	Operator	Meaning
Node	#this	the identifier of the current node
Conductor	#this	the identifier of the current conductor
	#nodeA	the identifier of the first node as defined in the conductor edit title
	#nodeB	the identifier of the second node as defined in the conductor edit title
Lump	#this	the identifier of the current lump
	#twin	the identifier of the twinned tank
Path	#this	the identifier of the current path
	#up	the identifier of the upstream lump
	#down	the identifier of the downstream lump
	#twin	the identifier of the twinned path
Tie	#this	the identifier of the current tie
	#node	the identifier of the attached node
	#lump	the identifier of the attached lump (not valid for HTUS, HTNS)
	#path	the identifier of the associated path (not valid for HTU, HTUS)
	#path2	the identifier of the second associated path (only valid for HTNC)
	#twin	the identifier of the twinned tie
Ftie	#this	the identifier of the current ftie
	#lumpC	the identifier of the first lump
	#lumpD	the identifier of the second lump
	#path	the identifier of the associated path (for AXIAL fties only)
	#twin	the identifier of the twinned ftie
Iface	#this	the identifier of the current iface
	#tankA	the identifier of tank A (referenced tank)
	#tankB	the identifier of tank B (the second tank)

Sinaps takes the indirect referencing a step further by allowing the user to create network based logic with indirect referencing. Please see "Indirect Referencing in Network Logic" on page 5-140 for details fo using indirect referencing in network logic.

6.2.4 Valid Expressions

When defining any expression it is important to define it in a manner that it is valid for all possible input values. Many processor variables cannot be initialized before certain processor analysis operations begin, but these variables are still valid references within input expressions. Examples of such variables include QTIE, DRLXCC, and TIMEN. Thus, these variables represent a bootstrapping problem: they do not have a valid value (and are normally initialized to zero) until the solution has proceeded far enough, yet input variables requiring those values to be set may have already been encountered and set. In most cases this does not represent a problem. However, consider the example of defining a source based on the value of TIMEN (the current problem time):

```
1, power/timen
```

Until TIMEN has been calculated in the first transient time step, the above expression will divide by zero and hence result in an arithmetic fault. Also, the source applied to that node in the first step will be undefined as long as TIMEN is undefined. Conditional expressions may be used as a remedy, and in the following case keeping the source zero (or at least a very small number) until TIMEN has been initialized, which might include preliminary steady-state runs:

```
1, power/((timen == 0)? 1.0e10 : timen)
```

6.2.5 Dynamic Registers

Registers are dynamic, which means they can be changed in the processor (i.e., during SINDA/FLUINT execution). The register table is simply a location for the user to define *initial* values. If the values are based on processor variables or are modified through user logic, the new register value will automatically be propagated throughout the model during the course of the solution. At any point during the solution (though more commonly within Output blocks), the user can write current values of registers to the output file using the call to the REGTAB subroutine:

```
CALL REGTAB                    $ write register names, values, and expressions to the output file
```

Changes made in logic will not be propagated until the current logic block has finished. Sometimes, the user will require an update to occur *within* the current logic block. For example, if a conductance has been defined as the expression “NuTurb*UpFact,” and if the user changes the value of UpFact in logic but needs the new conductor value updated immediately to enable subsequent operations, then the subroutine UPREG can be called immediately after setting the new register value to force the propagation of the value. This sequence is shown in the following example

```
UpFact = 1.1*UpFact
CALL UPREG                     $ propagate recent changes throughout the model
```

CHAPTER 7

Element Generation and Component Models

7.1 Macros: Element Generation and Component Models

Sinaps has various built in element generation tools. Some of these are unique to Sinaps such as the multi-conductor, others are based on SINDA/FLUINT macrocommands such as the CAPPMP, LINE, and HX macros. These tools assist the user in rapidly generating models by creating multiple elements, rather like a subnetwork, to simplify the creation of common components. Although some of these tools are based on the GEN commands in SINDA/FLUINT, in Sinaps some of these tools such as the "pipe", have been expanded. For example, the traditional LINE/HX macro has been expanded to include the option to create associated thermal nodes, and the ability to add exterior insulation.

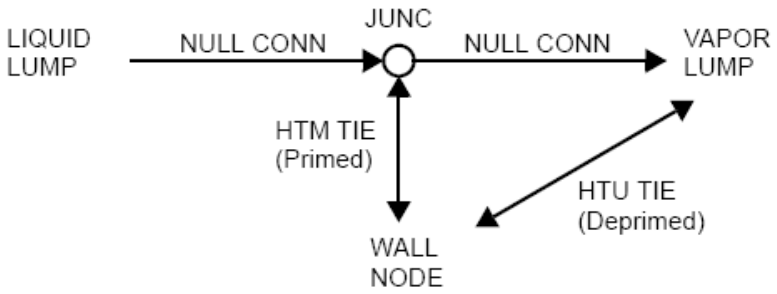
7.1.1 Capillary Evaporator Pumps (CAPPMP)

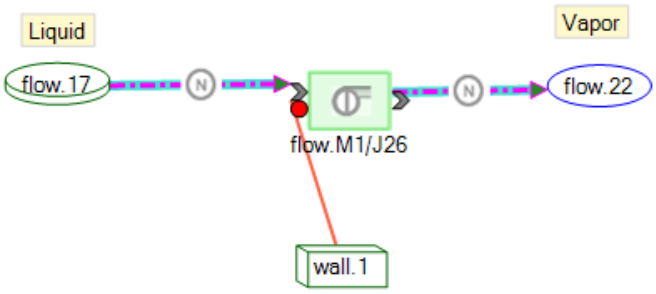
Capillary pumps, only applicable in two-phase single-constituent submodels, are used to model capillary pumping or condensation through a porous structure. This macro generates a junction between two NULL connectors: the simplest way to think of a CAPPMP model is a CAPIL connector with a junction in the middle. The presence of the junction allows heat to be added or removed from a CAPPMP model, whereas a CAPIL connector is adiabatic like any path. This junction has no other physical meaning, and the state of the junction is largely meaningless. If the heat rate into the junction is zero, the CAPPMP behaves exactly like a CAPIL connector. Unlike a CAPIL, the CAPPMP is direction sensitive.

If the heat load into the device is positive, the CAPPMP simulates an evaporator-pump. Liquid at the designated liquid end (i.e., not the designated vapor end) can be evaporated and forced into the designated vapor end, performing pumping by capillary action. If the pressure drop is too big or a capillary interface cannot otherwise exist, the device will deprime. This will generate error messages if the flow rate through the junction suddenly drops yet the heat load continues to apply. If the junction heat load is negative, any vapor present in the designated vapor side will be condensed, and any liquid will be allowed to pass to the liquid side if it is at a lower pressure.

Because the junction within a CAPPMP macro is really only a location into which heat is added or subtracted, and has no physical significance, the heat rate into the tie is not simply the product of the UA coefficient times the temperature difference between the node and the junction. When primed, the saturation temperature is used instead of the junction temperature. When deprimed, the tie will spontaneously jump to the designated vapor lump and become an HTU tie

with the same UA value. This usually eliminates the error messages that would result when a CAPPMP deprimes with a constant heat rate still being applied. If the CAPPMP reprimes, the tie will return to the central junction, and once again become an HTM tie. In either mode, the UA value can be modified. When tied, the CAPPMP macro can be schematically represented as (with path flowrates shown positive for evaporation):





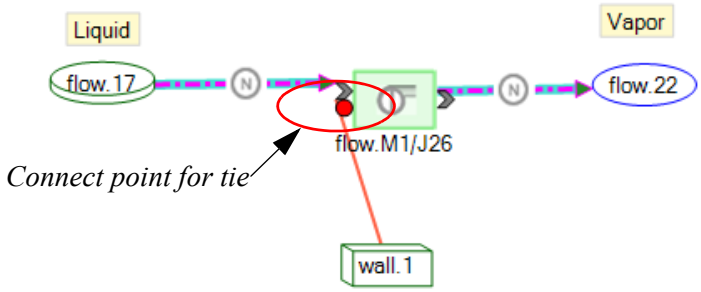
For more information on model capillary pumps, guidelines, restrictions, alternative, etc. please see Section 3.9.3 of the SINDA/FLUINT User's Manual.

7.1.1.1 Creating a Capillary Pump

To create a capillary pump, click the CAPPMP icon ( ) on the fluid toolbar and then click on the network to place the object. Alternatively, using the Tool Pallet, the icon can be dragged onto the network. Once placed on the network, the capillary pump must be connected both upstream and downstream to fluid lumps. The flow must come into the capillary pump on the left side of the icon; flow leaves on the right side. Once created, the device can be rotated and/or the flow reversed. The type of connector will automatically be set as NULL when created. Once created, these connectors cannot be edited by the user other than moving one end. Once created, the capillary pump must be edited to define the various input parameters.

A heat load can be applied to the capillary pump either by specifying a heat load (QL) directly to the internal junction, or by creating a tie between the junction and a thermal node (which is the preferred method). To create a tie, simply draw a link between the red dot of the capillary pump and a thermal node as shown by the red tie in Figure 7-1 below. No other ties may be added to a CAPPMP junction, and neither should fties be placed onto that junction (they should be applied instead to the endpoint lumps). Once the capillary pump tie has been created, it cannot be edited.

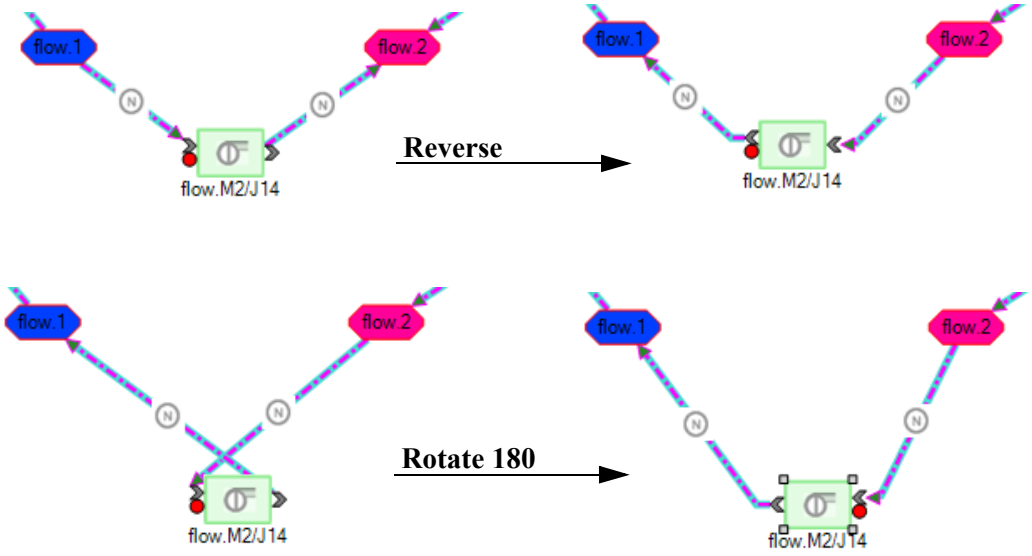
FIGURE 7-1. Applied a Heat Load to the Capillary Pump



The label of the capillary pump is in the format of [submodel].[macro ID]/[junction ID]. The label can be changed using the Object Properties dialog (see “Network Object Label” on page 4-9).

The orientation of direction of a capillary pump can be changed. After selecting a CAPPMP, clicking the right mouse button brings up the object edit menu. This menu allows the user to edit, move, delete, rotate 180,or reverse the macro. The reverse option reverses the flow through the macro while the rotate 180 option rotates the connect points on the macro as shown in Figure 7-2.

FIGURE 7-2. Rotating and Reversing a Capillary Pump



7.1.1.2 Editing a Capillary Pump

To edit a capillary pump, double-click on the object in the network window to open the Cappmp Edit dialog shown in Figure 7-3. The Data tab allows access to all the key parameters for the capillary pump. The Additional Input tab provides access to define text-based inputs for less commonly used parameters such as upper and lower limits on quality for a primed state, duplication factors, etc. For a list of all parameters and input syntax, see Section 3.9.3 of the SINDA/FLUINT User’s Manual.

Number - This is the unique macro ID. This number is automatically generated but can be overridden by the user.

Junction Number - This is ID number for the junction which will be created as part of the macro. This number is automatically generated but can be overridden by the user.

Temperature - This is the initial temperature for the junction to be created. This value will be defaulted if lump defaults have been defined.

Pressure - This is the initial pressure for the junction to be created. This value will be defaulted if lump defaults have been defined.

Quality - This is the initial quality for the junction to be created. This value will be defaulted if lump defaults have been defined.

Heat Load - This is the heat load applied to the junction of the capillary pump. In general it is recommend this be left as zero and the tie option be used, as discussed in “Creating a Capillary Pump” on page 7-2. If a tie has been created, this field will be grayed out.

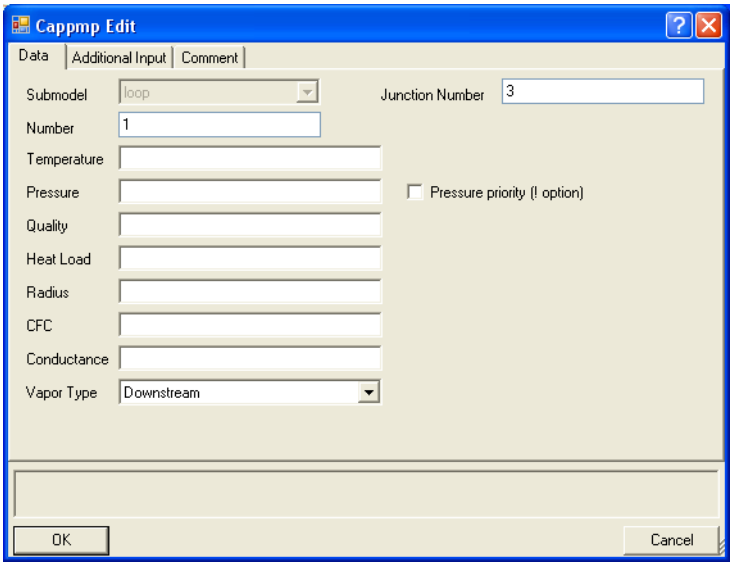
Radius (RC) - The effective capillary radius of the meniscus if it were two-dimensional. For circular tubules, RC is simply one half of the diameter. For a uniform (isotropic) wick, the capillary radius should be available from test data, vendor data, or estimates. For a thin slot, RC is equal to the slot width. Non-positive RC values are interpreted as perfect capillary devices capable of withstanding or producing an infinite pressure gradient. This is useful for preliminary design work or for controlling steady-state solutions to prevent deprime.

Capillary flow conductance (CFC) - CFC is the capillary flow conductance through the entire passage. See “Capillary Device (CAPIL)” on page 5-52 for more information on defining the flow conductance.

Conductance (UA) - The initial heat transfer conductance for the tie (if tied to a thermal node).

Vapor Lump - This pull-down menu identified which side of the capillary pump is the vapor end (the lump at which heat transfer may be assumed to occur). When a TIE option is used and the device deprimes, the tie jumps to the designated vapor endpoint lump and becomes an HTU tie. It returns to the central junction and becomes an HTM tie if it reprimes. No analogous movements of the heat load are possible if the QL option is used. Refer to Section 3.11.3 of the SINDA/FLUINT User’s Manual to learn more about primed/deprimed states.

FIGURE 7-3. Capillary Pump Edit Form



7.1.2 Continuous Flow Passages, Pipes, Ducts, and Macros

Duct macros in SINDA/FLUINT are very important modeling tools and are common to most flow models. There are two types of SINDA/FLUINT macros which are considered to be continuous ducts: Line and HX. The Line macro creates a series of lumps and paths identified as being a continuos flow passage, which will therefore include higher-order terms based on axial velocity gradients. The HX macro performs the same function as Line macro, but automatically creates heat transfer ties to a set of user specified nodes. The PIPE component (see "Pipes" on page 7-8) in Sinaps allows fast generation of either Line or HX macros. Additionally Sinaps has the ability scan a network and determine what portions of the network appear to be a duct and flag these sections as "automatically generated ducts" (see "Automatic Duct Generation" on page 7-6). Key information about pipes and SINDA/FLUINT macros is provided in this chapter, however for more in depth information on SINDA/FLUINT HX and Line macros, please see section 3.92 of the SINDA/FLUINT User's Manual.

There are important differences between lumps and paths generated or input separately and those generated in order to represent a duct with internal variations in properties due to heat transfer or side flow passages. The key to these differences is that lumps and paths within a SINDA/FLUINT macro, a Sinaps Pipe or automatically created duct, are known to represent a continuous flow passage, i.e. a duct. The use of any duct invokes additional calculations in SINDA/FLUINT to automatically calculate spatial accelerations due to heating, area changes, or side flow. Additionally, axial variations in velocity and the momentum transfer associated with phase change are accounted for. Causes for velocity gradients include density changes, area changes, side flow passages, and transient storage or release of mass in compressible tanks. Thus, ***pipes and automatically created ducts will respond differently than if the lump and paths had been input individually.*** The user should preferentially use these two options in Sinaps whenever modeling an axially subdivided flow passage.

Terminology clarification: The SINDA/FLUINT manual uses the terms HX/Line macro and duct macro interchangeably. In Sinaps, pipes, macros and ducts are distinctively different collections of network elements. To clarify, a SINDA/FLUINT ***HX or Line macro*** will import to Sinaps as a Pipe. A Sinaps ***Pipe*** is a higher level modeling object than an HX or Line macro, since it is capable of creating thermal network elements. In Sinaps, a ***duct*** is an means of automatically flagging a continuous series of paths and lumps to be treated as a Line macro during the SINDA/FLUINT solution process.

7.1.3 Automatic Duct Generation

To assist the user Sinaps automatically inspects each series of paths and lumps to determine if they represent a continuous flow passage. If the lumps are not plena, and if the paths are either tubes or Stubes with similar endpoint flow areas (AFI/AFJ or AF), similar diameters, and similar duplication factors, then the flow passage will be identified internally as a "***created duct***." These created ducts can be visually identified in the network using the network input check Show Created Ducts. This option will highlight each created duct in a different color. ("Checking for Created Ducts" on page 4-19.) Ducts can also be view in the Model Browser, see "Automatic Duct Tree" on page 8-9. All automatic ducts are created as "centered" Line macros.

Warning: Although Sinaps attempts to interpret what the analyst intends when it automatically aggregates elements into ducts, the human eye and knowledge of the model are the ultimate aggregation tool. ***It is highly recommended that the analyst verify proper aggregation of ducts in every model.***

A created duct can be as small as one lump and two paths and must always terminate at each end with a path, unless the aggregation produces an upstream or downstream duct. When multiple paths leave or enter a lump, Sinaps will look for the path with similar flow parameters for inclusion into a duct. If multiple paths exist (say one inlet and two outlets all with the same flow parameters), then the duct will terminate. However if only one exit path matches the inlet path, then the lump will be included and Sinaps will continue aggregating downstream elements into a duct (along that exit path).

Sinaps does not look for dominant side flow when aggregating ducts. It is the user's responsibility to inspect and verify created ducts. If dominant side flow is present and should break a duct, the user can use the element-level duct control feature to correct the aggregation, see "Excluding Specific Elements From Duct Generation" on page 7-7.

When aggregating ducts, Sinaps will aggregated multiple pipes if appropriate. However, *only one type of pipe* (centered, upstream, downstream, fill, or purge) can be aggregated into a single duct, i.e. a single duct can contain multiple centered pipes but it can not contain a centered pipe and an upstream pipe. Likewise fill/purge pipes will not aggregate with upstream/downstream pipes.

In general twinned lumps and paths are ignored by the aggregator with the exception of fill/purge pipes. If a fill or purge pipe is aggregated into a duct, Sinaps requires all other lumps and paths to match the twinned state of the pipe, i.e. twinning must be consistent for a fill or purge duct.

When paths and lumps are flagged by Sinaps to be part of a continuous duct, an additional parameter is sent to SINDA/FLUINT through the .sin file to define the flow passage as a SINDA/FLUINT Line macro. An example of this parameter is "MACRO = 1,9,1", which contains the information MACRO=[ID],[DUCT TYPE],[LOCATION]. Table 1 below summarizes the duct types. Duct types are interpreted by SINDA/FLUINT and used to set the discretization scheme used for the Line macro. Note that when a Sinaps Pipe is aggregated into a surrounding duct, the pipe will be designated as a Line macro regardless of whether it contains a wall or not: the ties will be defined separately. The MACRO information is designed to be a method of sending information between Sinaps and SINDA/FLUINT. Do not try to modify these parameters manual or to create them outside of Sinaps.

> Warning: The MACRO statement is a method of sending information between Sinaps and SINDA/FLUINT. This is not a user-definable parameter. Do not modify these parameters manually or create them outside of Sinaps.

TABLE 1. Autoduct Type Summary

Discretization	Line Macro	HX macro
upstream	4	7
downstream	5	8
centered	6	9

7.1.3.1 Excluding Specific Elements From Duct Generation

The user may disable portions of a created duct by unchecking the "Aggregate into continuous passage" check box on any or all of the paths or lumps in a created duct. The user may also use this option to force the definition of a duct when multiple paths with similar criteria enter or exit a lump; the user may uncheck the aggregation option in order to clarify ambiguities.

By definition, a Sinaps Pipe will always be a duct, and it might also be aggregated into adjacent ducts or other pipes if they meet the criteria stated above. To exclude a Sinaps Pipe from being collected as part of an adjacent pipe or created duct, edit the lumps adjacent to the pipe and uncheck the "Aggregate into continuous passage" check box.

7.1.3.2 Disabling Automatic Duct Generation

The automatic duct generation can be globally disabled for a model using the Automatic Duct Generation toggle under the Edit menu in the network window. This automatic duct toggle is independent of the path and edit form Aggregate into Continuous Passage option. The path/lump edit form options specifically flag a single path or lump to be excluded from automatic ducts while the toggle works globally on the model to *disable all* automatic duct generation. When the auto generation is toggle on, the "Aggregate into continuous passage" options take effect.

> Note: It is recommended the user *not* rely on the automatic duct generation when creating networks. It is best if the user determines where macros should be and initially creates the network using the appropriate Pipe (wall or no wall) option.

7.2 Pipes

In Sinaps, the *Pipe* component allows the generation of both Line and HX macros (see "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5), however the pipe object takes the component generation a step further by providing for the automatic generation of associated wall nodes and conductors and insulation to be added to the outermost surface of the pipe wall. When creating a pipe, options are provided for creating a pipe with a wall, a pipe without a wall, and a "wall only" pipe.

Pipes are used for modeling a continuous flow passage with or without a wall. Pipes can be defined based on standard pipe tables, by user-definable pipe libraries, or through dimensional inputs. If a pipe is defined as a Pipe with wall, the wall material is defined using the Sinaps thermophysical property database (see "Thermophysical Property Manager" on page 5-142). Walls can be define with up to two layers of nodes radially and may include up to 10 layers of insulation on the outside surface. Pipes are akin to collections in that all the network elements created by the pipe will appear in a box. To distinguish the pipe from a collection, the pipe box is colored green, whereas the collection has no fill color.

For a complete understanding of pipes and their usage, please also read "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5.

7.2.1 Types of Pipes

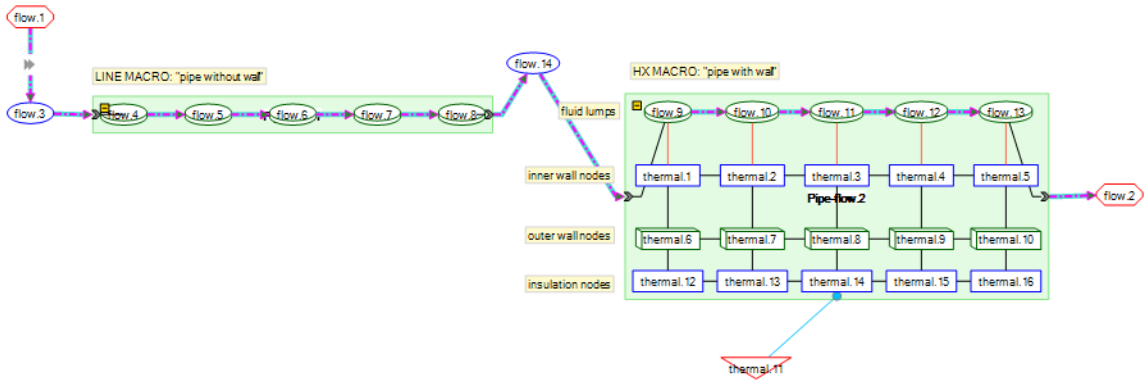
The *Pipe without wall* option creates lumps and paths based on user-defined geometry for the pipe. Beyond the standard SINDA/FLUINT Line macro, this Pipe also includes the ability to add wall roughness to the pipe wall for frictional pressure drop calculations and the ability to enable fill/purge modeling.

The *Pipe with wall* option creates the entire network require for a fluid pipe including nodes, conductors, fluid lumps, paths, and ties. This option goes beyond the basic SINDA/FLUINT HX macro with the automatic generation of nodes and both axial and radial conductors through the wall based on the wall dimensions, along with the ability to apply insulation to the outermost nodes of the pipe.

Figure 7-4 shows pipes representing Line (pipe without wall, on the left) and HX (pipe with wall, on the right) macro in a sample network. The pipes are shown in the expanded view to display the complete subnetwork. Only the elements inside the green box are part of the pipe.

> Note: When an HX macro is imported from either a SINDA/FLUINT input file or from a Sinaps*Plus* export file, the HX macro will only include the fluid components: the lumps, paths and ties, but will exclude the nodes.

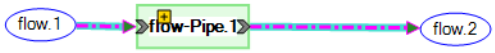
FIGURE 7-4. Example of LINE and HX Macros



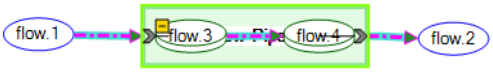
The "*Wall only pipe (no fluid)"* option creates the network of the pipe wall. No fluid elements are created. This option can be useful for modeling an empty cylinder or a solid rod (or wire), where insulation may be present or there is a need to connect the exterior of the pipe/insulation to an environment via a multi-conductor element (Section 7.3). When using this option, be aware that pipes are defined as a fluid modeling object, therefore the master model must contain at least one fluid submodel in order to create a wall-only pipe. If there are no fluid submodels present, a fluid submodel must be created and at least one plenum must be added to the submodel to make it a valid submodel (default fluid properties can be used when creating the submodel).

7.2.2 Creating a Pipe

To create a Pipe click on the "Pipe" icon ( ) and then right-click on the network diagram to drop the pipe onto the diagram. Alternatively, drag the "Pipe" icon from the Tool Pallet to the network diagram. After the pipe icon is placed on the diagram, draw links to connect it to the network. The paths will default to Stubes. These can be changed to tubes by editing the pipe. Once the pipe is placed on the diagram, it must be edited to complete the definition process. Click on the [+] in the upper left corner of the pipe to expand the pipe network. By default, when a pipe is created it will be created in the current submodels and on the current layer of the network.



Pipe As Created



Expanded Pipe Network

Pipe Discretization: The Pipe has three options corresponding to the SINDA/FLUINT macro discretization schemes, depicted in Figure 7-5:

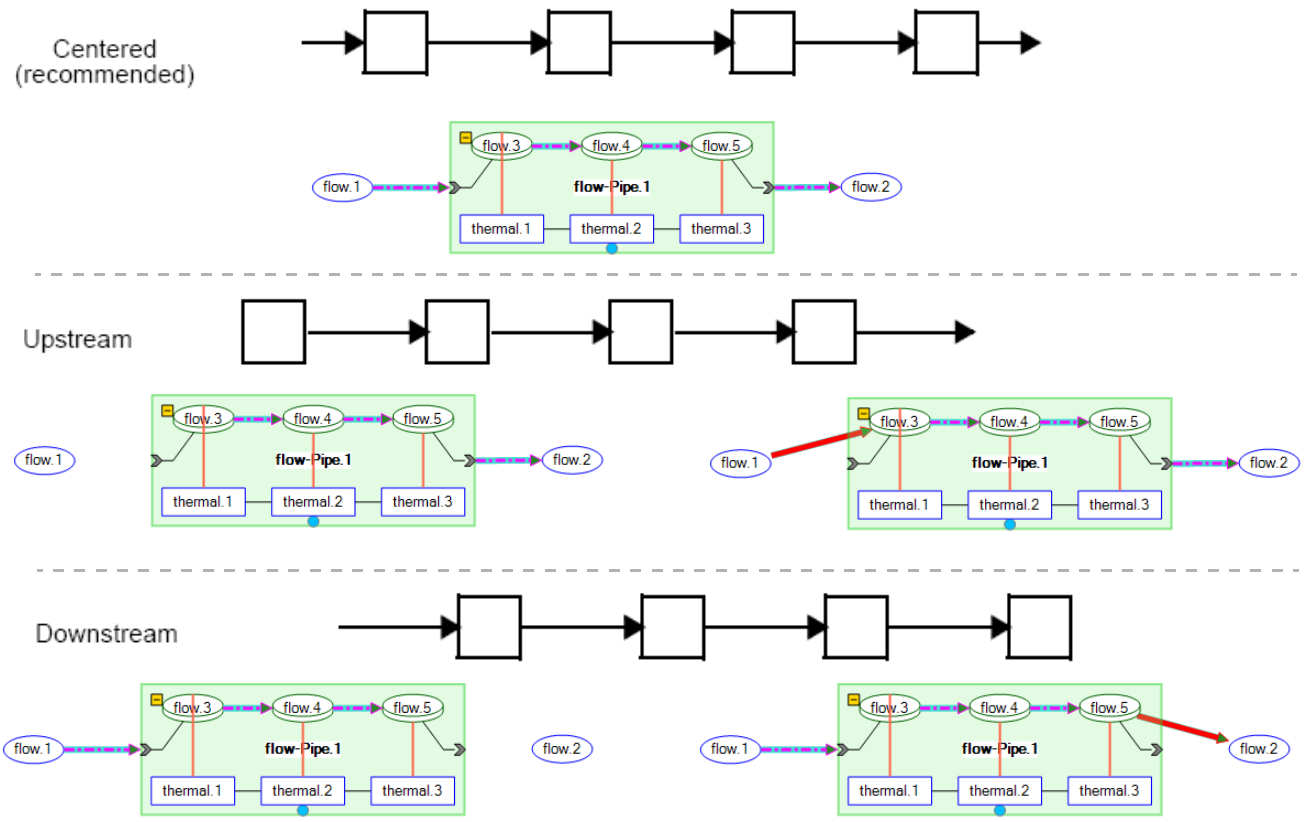
Upstream Option . . . . .Generates pairs of lumps and paths in series: lump —> path —> lump —>path... that model a line or pipe. Lumps are upstream of the corresponding path. Paths are tubes or STUBE connectors. Lumps are tanks or junctions. Except when using flat front purge modeling, this option is discouraged.

Downstream Option . . . . .Generates pairs of paths and lumps in series: path —> lump —> path —> lump ... (Same as U option but lumps are at downstream end of each section.) Except when using flat front fill or priming modeling, this option is discouraged.

Centered Option . . . . .Generates a series of paths and lumps that begin and terminate with paths: path —> lump...(more path/lump pairs)... —> path. (Lumps are centered between corresponding paths.)

These discretization options are automatically invoked depending on how the user connects the pipe to the network. Connection options are shown in Figure 7-5. If the user connects the pipe via both connect points to the network, a center pipe will be created. If the pipe is only connected to the network at the downstream end, the upstream option will be invoked. Likewise, a connection made only at the upstream end of the pipe will invoke the downstream option. If the user wishes to create an upstream or downstream pipe, but needs to connect both ends of the pipe to the network, the pipe can be forced to be upstream or downstream by *connecting directly to the end lump* within the pipe instead of the pipe connect point.

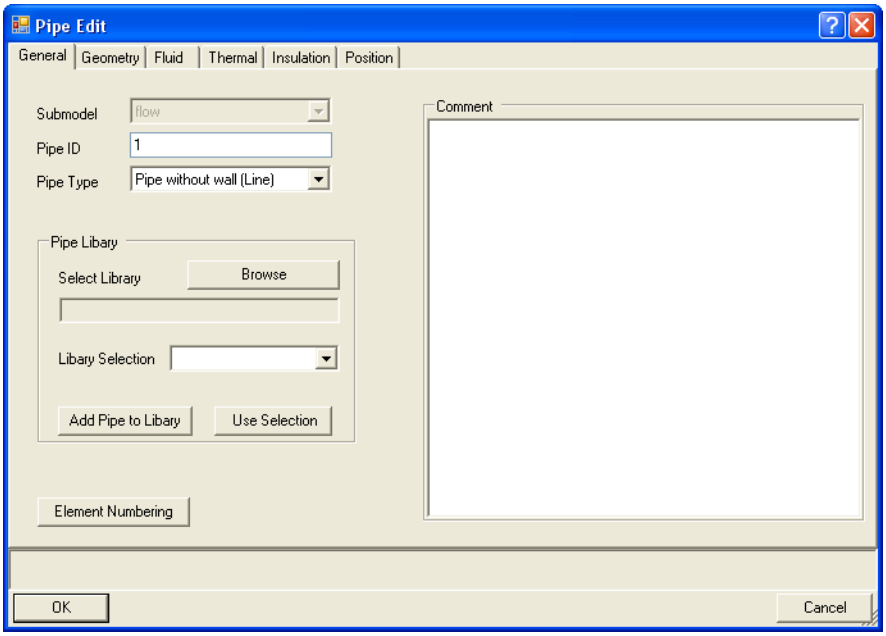
FIGURE 7-5. Pipe Discretization Options



7.2.3 Editing a Pipe

To edit a Pipe, double click on the frame of the pipe collection to open the Pipe Edit dialog shown in Figure 7-6. Alternatively the pipe can be edited by right-clicking on the pipe frame and selecting Edit Pipe. This multiple tabbed form groups inputs by topic. Click “OK” after making all changes to exit the form. The contents of each tab are addressed herein. For detailed general information on pipes, types of pipes, and connecting to pipes, see section “Pipes” on page 7-8. Multiple pipes can be edited collectively through the model browser, see “Editing Multiple Pipes” on page 7-23.

FIGURE 7-6. **Pipe Edit Form**



Pipe General. The General tab contains data fields for defining top-level parameters for the pipe. The comment field allows the user to provide a comment for documentation purposes. The Edit Elements tab provides data fields for inputting various element-specific parameters.

Pipe ID - This is the identification number for the pipe. Automatic numbering is provided however the number can be changed. This number is used as the reference number for a pipe in the model browser. Each pipe must have a unique identification number.

> Note: By default pipes are labeled based on their ID and the submodel in which they reside. The default label for a pipe is [submodel name]-Pipe.[Pipe ID], for example Flow-Pipe.3. The user can override the label using the Object Properties dialog.

Pipe Type - This is type of pipe to be created. Options consist of a Pipe without wall (fluid only), Pipe with wall, or Wall only pipe. Based on the pipe type selection, some portions of the pipe edit form will become inactive (indicated by being gray) if they are not applicable to the type of pipe selected.

Comment - The comment field allows the user to provide documentation on a pipe. This could be a reference part number or system location information.

Pipe Library - The pipe library allows the creation of a user-defined custom pipe library. Library files are in xml format. The library saves all information in the pipe form with the exception of the length, resolution, duplication factors, network logic, and position data. Note that parameters not controlled by the pipe form will not be saved to the library, e.g. K factors, tank compliance, etc. Once a pipe has been created in Sinaps it may be added to the pipe library for future use. Once in the library, the pipe information can be retrieved and used to populate other pipe edit forms. The

resulting pipe library can be accessed and used with any model (including those with different units). It is the user's responsibility to define library locations, library names, and to control access to the library. If sharing a library, be aware that other users can add, redefine, and delete entries in the pipe library.

> Note: Library items are unit specific and will not be converted if imported to a model with different units (SI or ENG). Library items are saved with a unit flag indicating (SI or ENG). if a library item is imported into a model with different units, a message will appear stating the units do not match,

Select Library - Use this button to browse to, or create, a library file (*.xml format). When creating a new library file, click "Browse" to open browser window, browse to a location for the new library file, and type a name into the "File Name" field, e.g. pipe_library or pipe_library.xml. This file can be saved locally or on a network server. Once a library has been selected, the path and name of the library will appear in the associated field below the browse button.

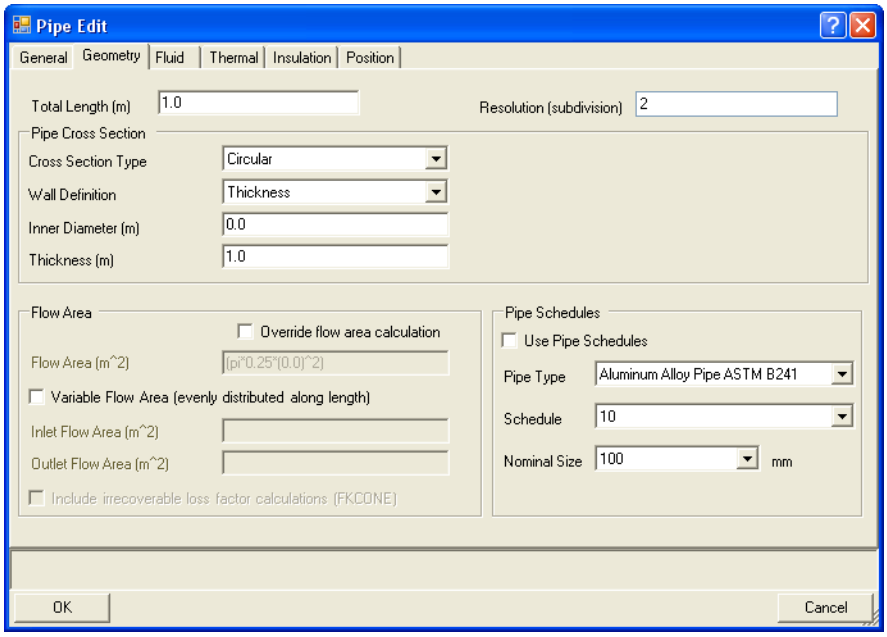
Library Selection - This field provides a pull-down list of all pipes defined in the selected library. If the library is empty, no entries will be listed.

Add Pipe to Library - Use the Add Pipe to Library button to add the current pipe description to the library. When prompted for a name, provide a unique name for the pipe. The current configuration of the pipe will be saved excluding the length and resolution. Subsequent changes to the pipe in the model will *not* change the saved library item. To update a pipe in the library, after editing the pipe, return to the pipe library dialog and select the appropriate pipe name from the library and use the Add Pipe button to save the updated pipe to the library under the same name. For a different name, simply provide a new name when prompted with the Pipe Name dialog and both prior and updated pipe descriptions will be available in the library.

Use Selection - This button populates the pipe form with information that had been saved in the pipe library. After a pipe library has been selected and a pipe selection made in the Library Selection field, click the Use Selection button to apply the pipe library item to the open pipe edit form. All fields are imported with the exception of length, resolution, and duplication factors. The pipe library will save the material name for the wall. If the material does not exist in the current thermophysical properties database, a pop up message will appear prompting the user to go to the thermal tab of the pipe form and add a material.

Element Numbering - The element numbering button brings up the element numbering dialog, allowing the user to use new integer identifiers. This dialog allows the user to renumber all elements in a pipe by specifying the starting number and the increment. A check box allows each element type to be excluded from the renumbering process. For more information see "Renumbering a Pipe" on page 7-25.

Pipe Geometry. The Pipe Geometry tab contains data fields for defining the pipe geometry such as diameters, wall thicknesses, and flow areas.



Total Length - This is the total length for the Pipe.

Resolution - This is the axial resolution of the pipe: the number of lumps to be created along the length of the pipe. Note that once a pipe has been created, if the resolution is edited and expanded, the parameters of the first elements (e.g.lumps, nodes, etc.) in the pipe will be used for setting parameters of the new elements. If customizations such as k-factors had previously been made to the "first" elements, these *may* be brought to the new elements. For this reason is it recommended elements in pipes not be customized.

Pipe Cross Section - This group of data fields defines the geometry of the pipe. The primary fields are Cross Section Type and Wall Definition. The remaining (secondary) fields are dynamic and will vary depending on the input selected for the two primary fields and the Pipe Type on the General tab. Table 2 provides a summary of the various input options. Details of each input field are provided herein.

TABLE 2. Summary of Pipe With Wall Input Options

Cross Section	Wall Definition	Input Fields	Comments
Circular	Thickness	ID, thickness	
Circular	Mass/Length	ID, mass/length, perimeter	limited to one radial node, does not support external insulation
Circular	Cross Sec Area	ID, cross sectional area, perimeter	limited to one radial node, does not support external insulation
Circular	Standard	OD, thickness	

TABLE 2. Summary of Pipe With Wall Input Options

Cross Section	Wall Definition	Input Fields	Comments
Circular	User defined	ID, perimeter	limited to one radial node, does not support external insulation
			Pipes defined as "circular flow area" in versions prior to 5.3 will be migrated as Circular/User defined.
Rectangular	Thickness	width, height, thickness	does not support external insulation
Rectangular	Mass/Length	width, height, mass/length, perimeter	limited to one radial node, does not support external insulation
Rectangular	Cross Sec Area	width, height, cross sectional area, perimeter	limited to one radial node, does not support external insulation
User Defined	User Defined	hydraulic diameter, optional flow area	Limited to one radial node, does not support external insulation.
		use edit element buttons on thermal and fluid tabs to define necessary input parameters	Pipes defined as "user defined area" in versions prior to 5.3 will be migrated using this option.
			Pipes imported from text files or SinapsPlus exports will be imported using this option

Cross Section Type - This field defines the cross section of the flow area. Three options are available: Circular, Rectangular, and User Defined. For "Circular," the flow area is based on the diameter, whereas rectangular pipes use the product of the height and width for the flow area. "User Defined" pipes require a hydraulic diameter and flow area if non-circular (flow area defaulted as circular using the hydraulic diameter, can be overridden in the Flow Area portion of the form).

Wall Definition - This field defines how the wall geometry will be defined. Pull-down options will vary based on the Cross Section Type selection above. For "Circular" pipes the options include Thickness, Mass/Length, Cross Sectional Area, User Defined, and Standard. The Thickness option provides the ability to define a pipe based on inner diameter and wall thickness, whereas the Standard option allows the pipe to be defined using outer diameter and wall thickness. For "Rectangular" pipes, the options include Thickness, Mass/Length, and Cross Sectional Area. For "User Defined" pipes, the user must use the element edit buttons on the Fluid and Thermal tabs to define all required input fields.

Hydraulic Diameter - This is the hydraulic diameter (DH) for a user-defined pipe. The hydraulic diameter is defined as four times the flow area divided by the wetted perimeter. This reduces to the diameter for a circular cross section, and to the length of one side for a square cross section.

> Note: If the Variable Flow Area option is selected (AFI and AFJ are input), the hydraulic diameter is an optional input. If no value is input, DH (and DEFF) is initialized as the diameter at the average flow area assuming a circular cross section.

Inner Diameter - This is the inner diameter of a circular pipe. The value is interpreted as the hydraulic diameter.

Outer Diameter - The outer wall diameter of a circular pipe. This field is applicable for circular pipes with "standard" wall definition.

Inner Width - This is the width of the flow channel for a rectangular pipe.

Inner Height - This is the height of the flow channel for a rectangular pipe.

Thickness - The wall thickness for either a circular or rectangular pipe.

Pipe Perimeter - The outer perimeter of a pipe. This field is applicable for a pipe with wall if it has a user-defined cross section or if it has the wall definition set as mass per length, cross sectional area or user defined. The perimeter is used in determining surface area for contact when connecting to the outer pipe wall using a multi-conductor link (see "Multi-conductor Element" on page 7-27). Even if a multi-conductor is not connected to the pipe, the field must be non-zero.

Mass/Length - This is the mass per unit length of the pipe wall. Applicable for circular or rectangular walled pipes. The mass/length is used to calculate wall node capacitance and axial conductance. Pipes defined with mass/length as the wall definition are limited to single radial node (meaning the distinction between inner and outer wall temperatures--the temperature difference through the wall--is neglected).

> Warning: When defining a pipe wall using the mass/length option, the density portion of the wall node capacitance term is based on the mass/length input, not the thermophysical property database. It is the user's responsibility to insure the mass/length field and the material property selection on the thermal tab of the form are correctly matched.

Cross Sec. Area - The cross sectional area of the pipe wall. This field is applicable for circular or rectangular walled pipes. The cross sectional area is used to calculate wall node capacitance and axial conductance. Pipes defined the cross sectional area as the wall definition are limited to a single radial node (meaning the distinction between inner and outer wall temperatures--the temperature difference through the wall--is neglected).

User Defined - This option allows the user to specify a custom wall pipe. It is applicable primarily to "user defined" cross section types. However, it also exists as an option for circular pipes to support migration of pipes from early versions of Sinaps. When the User Defined option is used, the element edit buttons on the fluid and thermal tabs should be used to set up the various network elements.

> Warning: If the pipe has been imported from text based SINDA/FLUINT input files or from Sinaps*Plus* V4.8 or earlier, it will be treated as a User Defined cross section and User Defined wall definition. The pipe elements will be flagged as customized on the fluid and thermal tabs. Changing any field in the pipe form for a customized pipe will result in all customizations being lost. It is therefore recommended that such an imported pipe be redefined using the new pipe form to allow future editing.

Flow Area Section - This section of the form allows the user to customize the flow area of the pipe.

Override Flow Area Calculation - This check box can be used to override the default flow area. The default flow area is automatically calculated based on the inner diameter or hydraulic diameter, assuming the pipe is circular or the width and height if the pipe is rectangular. For a user-defined flow cross-section, the flow area will default to circular and it is the user's responsibility to override as necessary.When the box is checked the Flow Area field will become active and the new area can be defined.

Flow Area - This is the cross-sectional flow area for the paths (if twinned, it applies to whole passage). It is only applicable if the Override Flow Area box (above) is checked. Also note that greyed (inaccessible) expressions visible in this field (when the override box has not been checked) represents the current value or expression that will used by Sinaps.

Variable Flow Area - This option is only available for pipes *without* walls. By default, the flow area is assumed to be constant along the length of a pipe. This box can be checked to enable axially varying flow area. Once checked, the Inlet Flow Area and Outlet Flow Area fields will become active. The variation in geometry will be evenly distributed along all lumps and paths within the pipe. If the goal is to model a step change in the pipe diameter, two pipes of different diameter should be created with a reducer/expander path located between the pipes the added irrecoverable loss or K-factor).

> Tip: A modeling trick to add the effects of a an expander or reducer to the end paths of a pipe (with or without a wall) or any path within the pipe, is to edit the individual paths and defined AFI and/or AFJ on the Additional Input tab. This will not result in the pipe being toggled to a customized state, yet it will pass the AFI/AFJ values to the processor.

Inlet and Outlet flow areas - If the cross sectional area varies axially along the length of the pipe (i.e., the above box has been checked), the user can specify the inlet and outlet flow areas. The change in flow area will be evenly distributed across all the paths/lumps of the pipe. Variable geometry is *not* applied to the wall thermal nodes and conductors.

Include irrecoverable loss factor calculations (FKCONE) - This check box signals Sinaps to include calls to the SINDA/FLUINT subroutine FKCONE for calculations of the irrecoverable loss factors for conical reducers and expanders, per Section 3.15.2.4 of the SINDA/FLUINT User's Manual.

Pipe Schedules Section - Sinaps provides several built-in pipe size/schedule tables. This option specifies a pipe by the standard type, schedule number and nominal size. Based on the selection, Sinaps automatically sets the cross section type to circular with a standard wall definition and the outer diameter and the thickness are pulled from the built-in tables. When using the pipe schedules, the table must be selected first, followed by the schedule, followed by the size.

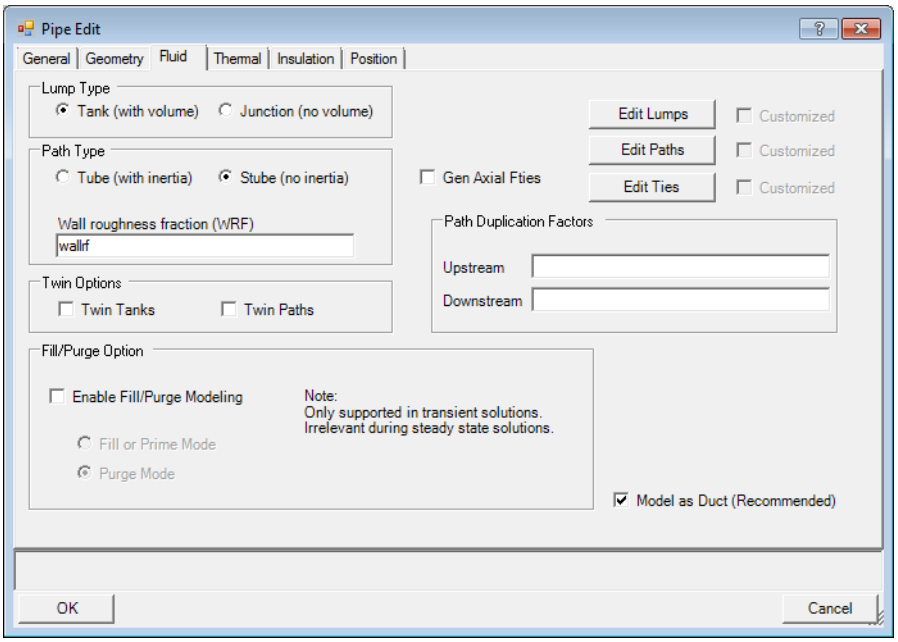
Use Pipe Schedules - To active the pipe schedule dialog, this box must be checked.

Pipe Type - Select the standard pipe table to use. Options include aluminum alloy pipe ASTM B201, B347, and B345, ANSI B36.10-75 sanf B36.19-65, ASTM D-1785, AWWA C900-89, CRANE Technical Paper 410, Handbook of Cast Iron Pipe, seamless copper pipe ASTM B42-66 and ANSI H23.16-67, and threadless copper tube ASTM B302

Schedule - This is a pull-down listing of available schedules (usually affecting pipe thickness).

Nominal Size - This is a pull-down listing of available nominal sizes.

Fluid Tab. The Fluid tab contains inputs related to setting up the fluid portion of the pipe. These options include defining network elements types (Tank vs. Junction, Tube vs. Stube), twin options, generation of axial fties, defining a wall roughness fraction, and setting up fill/purge modeling options.



Lump Type - Select the type of lump to be used throughout the pipe. Junctions have no volume and respond instantaneously whereas tanks have finite volume (and therefore take a finite time in a transient to heat up or cool down, fill or empty, etc.). For detailed information on tanks versus junctions see “Fluid Lumps” on page 5-23.

Path Type - Select the type of path to be used through the pipe. Pipes can only have tubes or “Stubes” (short tubes). Tubes have inertia whereas Stubes respond instantaneously. For more information on tubes and Stubes see “Fluid Paths” on page 5-36.

Wall Roughness Fraction (WRF) - The roughness fraction is defined as the ratio of the characteristic length of the wall roughness to DEFF (usually the same as DH), and is usually on the order of 1.0^{-6} to 1.0^{-3}. If no WRF is given, it is assumed to be zero (smooth wall—this does not mean frictionless). WRF is available in the user logic blocks and may be altered if necessary. WRF is only relevant if IPDC is nonzero and does not affect friction when the flow is laminar.

Twin Options - If the user wishes to model two-phase slip flow in the pipe, all paths including the inlet and outlet paths of the pipe can be twinned by checking the Twin Paths option. To invoke phasic nonequilibrium methods, both the tanks and paths can be twinned by selecting the Twin Tanks option. When pipe tanks are twinned, paths must also be twinned, as well as any heat transfer ties applied to those tanks.

When a non-twinned (homogeneous flow) pipe is connected to twinned lumps outside of the pipe, the paths will be connected to the liquid side by default. The user may override this assumption by changing the Lump Dn L/V and the Lump Up L/V path options through the Object Properties dialog.

Gen Axial Fties - This option allows axial Fties to be created parallel to all paths in the pipe. This option allows the inclusion of heat conduction through the fluid along the length of the pipe. Fties are generally only relevant in very slow moving flows or with highly conductive liquids. The generated Fties will have the same identifier as the paths. If the endpoint lumps are twinned and so is the path, a pair of twinned fties will automatically be generated using this directive. The use of this ftie option is preferred over independent ftie generation since it will automatically follow rules based on the twin state of the associated lumps and paths.

Dup Factors Upstream/Downstream - Pipes may be assigned duplication factors, which is a general device for exploiting symmetry to reduce the size of the model. The key to understanding duplication factors is this: the defined upstream lump "sees" a pipe DUPI times, while the defined downstream lump "sees" the same pipe DUPJ times. For example, an upstream duplication factor (DUPI) of 10.0 means that 10.0 such pipes appear from the perspective of the defined upstream lump even though only one pipe exists. Upstream and downstream duplication factors do not need to be equal. Duplication factors do not need to be integers. Zero values are commonly used to "shut off" a pipe, which causes the pipe to be ignored from the viewpoint of the endpoint lumps, even though the pipes's flow rate will continue to be updated. For more information on duplication factors, see section 3.12 "Symmetry and Duplication Options" in the SINDA/FLUINT User's Manual.

Enable Fill/Purge Modeling - This option enables modeling the progression of a liquid or vapor front through a pipe (flat front modeling). The user must specify whether the pipe is in a fill or a purge mode for the positive flow direction. When invoked, SINDA/FLUINT will set path STAT, IPDC, and UPF factors. When invoked, Sinaps will automatically change the fluid elements to tanks and tubes and will convert the pipe from centered to either up- or downstream discretization. Twinned tanks are generally recommended for flat front modeling, and these in turn require twinned paths and ties (which will be chosen automatically). For detailed information on flat front modeling, see section 3.27 in the SINDA/FLUINT User's Manual.

> Warning: Flat-front modeling is not a general-purpose option to be used with any class of two-phase flow. Rather, it is intended to handle a specific "degenerate" class of problems for which the extreme assumptions implicit in the flat-front approach can be made.

Edit Lumps/Paths/Ties - The element edit buttons (lumps, paths, and ties) allow the user to set additional parameters *beyond those which are controlled via the pipe form* (e.g., initial temperature, loss factors). Note the position data tab in the Edit Lumps dialog will be overridden data in the Position tab of the Pipe form.

Customized - When checked, these boxes signal that lumps, paths, or ties have been modified ("customized") outside of the pipe form, possibly through the element edit form or object properties. The box will *only* become checked if one of the parameters normally controlled by the pipe form has been modified such as lump volume, path diameter, wall roughness, etc. If a network element is modified for a parameter not controlled by the pipe form, such as adding a loss factor, the customize button will not toggle to a checked state.

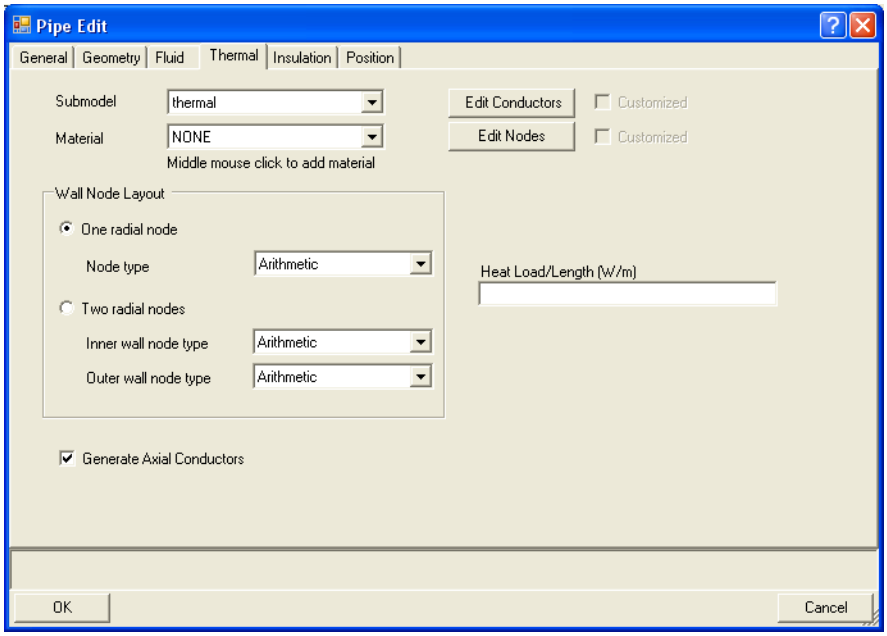
Once an element has been "customized" outside of the pipe form, the pipe edit form can no longer be used to modify pipe level parameters without resetting them all (perhaps losing customizations). If used, clicking "OK" in the pipe form will overwrite all customizations of pipe level parameters. If "OK" is clicked on a customized pipe, a message will appear stating the pipe has been customized. In this case, the user can inspect the fluid and thermal tabs of the pipe

form to see which type of element has been customized. If elements in a pipe are customized, the pipe comment field on the general tab is a useful tool for documenting all customizations for future reference.

> Caution: Avoid customizing pipe elements outside of the pipe edit form. Once customized, pipe level edits become blocked and must be overridden. Affected values include changed the pipe length, diameters, resolution and element type, just to name a few. Instead of customizing a pipe, it is recommended that multiple pipes be created to model any axial variations such as changes in diameter, flow area, wall thickness or material, or insulation properties.

Model as Duct - This checkbox enables/disables duct modeling at the pipe level. It is recommended this feature be enabled to fully capture accelerations associated with density and area changes, transient storage, and side passages. If this feature is disabled, it is the user's responsibility to add logic or expressions for these calculations. For more information about duct modeling, please see "Continuous Flow Passages, Pipes, Ducts, and Macros" on page 7-5.

Thermal Tab. The Thermal tab contains inputs related to setting up the thermal portion of the pipe: the model of the wall. These options include specifying a thermal submodel for the wall network, defining network element types (e.g., arithmetic vs. diffusion nodes), defining radial resolution of the wall, adding a heat load along the length of the pipe, and choosing whether or not axial conductors will be generating between nodes (they are generated by default).



Submodel - Select the submodel in which the thermal network elements will reside. This field defaults to the current thermal submodel in Sinaps.

Material - Pipes use the material database for thermophysical properties, so a material within that database must be selected using the pull down list. If a new material is to be added to the database, this can be done from the pipe form

by placing the curser in the material data field, click the middle mouse button, and select Add Material. The dialog for adding a material to the database will appear. For more information on defining materials in the database see “Thermo-physical Property Manager” on page 5-142.

Wall Node Layout - Sinaps allows pipe wall to be defined with either one node radially through the wall or two nodes. Use of a single node is appropriate if the pipe material is highly conductive and/or the thickness small, such that the temperature drop across the pipe wall can be neglected in exchange for faster solution speeds. Using this option, the user selects either One radial node or Two radial nodes and then defines the type of nodes as either arithmetic (no mass) or diffusion (with mass). For more information about node types see “Thermal Nodes” on page 5-9.

> Caution: In cases where the pipe contains fluid and the wall conduction is high with little or no gradient expected through the wall, the use of two radial wall nodes is discouraged. When a significant gradient is expected, using two radial wall nodes in a pipe to capture the effects of wall mass, it is often a better modeling practice to set the interior node to arithmetic and the outer node to diffusion. Since the inner node becomes tightly tied to both the fluid lump and the outer thermal node, this method allows the inner node to “float” between the two. If the inner node is a diffusion node, the fluid and outer node will ultimately “fight” each other for control over the inner node, possibly resulting in smaller time steps and longer execution time. See“Heat Transfer Ties” on page 5-101 for more information.

Heat Load/Length - Using this field a heat load/loss per length can be applied to the exterior wall of a pipe. The input should be in the form of power per unit length, with a negative value representing a heat loss. Based on the length of each node, Sinaps will automatically calculated a heat source term on each node which will be added in the source data block when the model is launched. When a wall is defined using two radial nodes, the load will be placed on the outer wall node. *If the pipe has insulation on it, the heat load will remain on the wall node, not on the insulation node.* This feature is useful for specifying a heat leak per length for large vacuum-jacketed cryogenic lines, guard heaters, or any other type of parasitic heat gain or loss. This option is less suited for modeling small vacuum jackets.

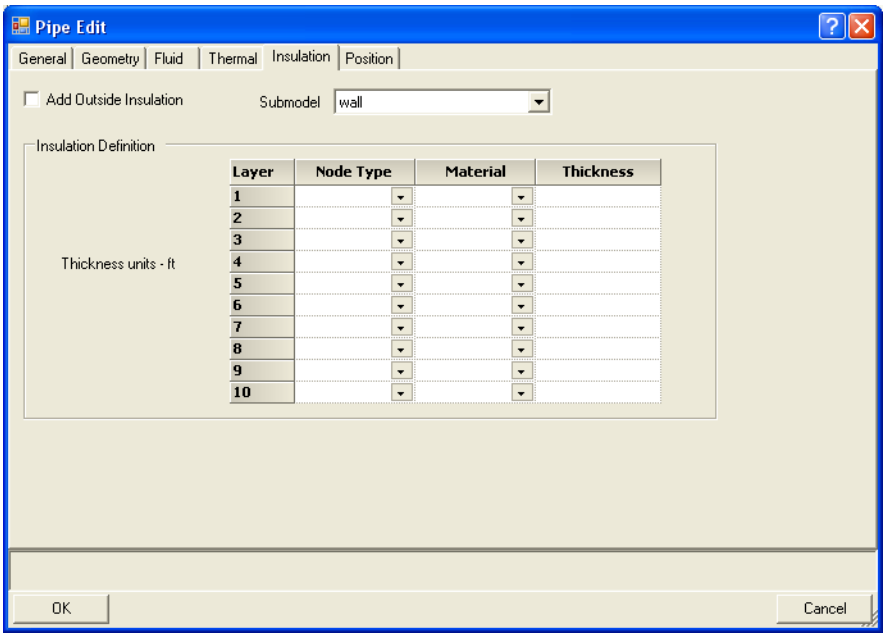
Generate Axial Conductors - By default Sinaps will generate axial conductors between nodes along the length of a pipe. The Generate Axial Conductors check box can be unchecked to disable this feature resulting in no axial conductors being generated. If check axial conductors will be used in the pipe wall and insulation if included with the pipe. When unchecked, all axial conductors will be removed from the pipe wall and insulation.

Edit Node/Conductors - The element edit buttons (nodes and conductors) allow the user to set additional parameters beyond those which are controlled via the pipe form (e.g., temperature, loss factors). If the wall definition has been set to User Defined, these edit buttons can be used to define the nodal capacitance and axial conductance values.

Customized - When checked, these boxes signal that either nodes or conductors have been modified (“customized”) outside of the pipe form, possibly through the element edit form or object properties. The customized box is not applicable if “user defined” is selected as the *wall definition*. The box will *only* become checked if one of the parameters normally controlled by the pipe form has been modified such as capacitance or conductance. If a network element is modified for a parameter not controlled by the pipe form, such adding a source term to a node, the customize button will not toggle to a checked state. For more information on customizing pipes, see “Customizing a Pipe” on page 7-26.

Insulation Tab. The Insulation tab allows insulation to be added to the outside of a pipe. (This option is only available for circular pipes whose outer diameter is known.) Up to ten layers of insulation can be defined, each using per-

haps distinct materials and thicknesses. Each layer is limited to one node radially, although multiple layers can be defined using the same material to capture temperature gradients within the insulation. Once added to a pipe, the thermal network representing the insulation nodes and conductors will be visible in the pipe collection. Options are available to suppress the visibility of insulation on a pipe.* When suppressed, the background of the pipe collection will turn pink as a visual indication that insulation is present.



Add Outside Insulation - This is a toggle to turn the outside insulation on or off. The box must be checked to enable insulation.

Submodel - Select the submodel the thermal network elements will reside in. This field defaults to the current thermal submodel in Sinaps.

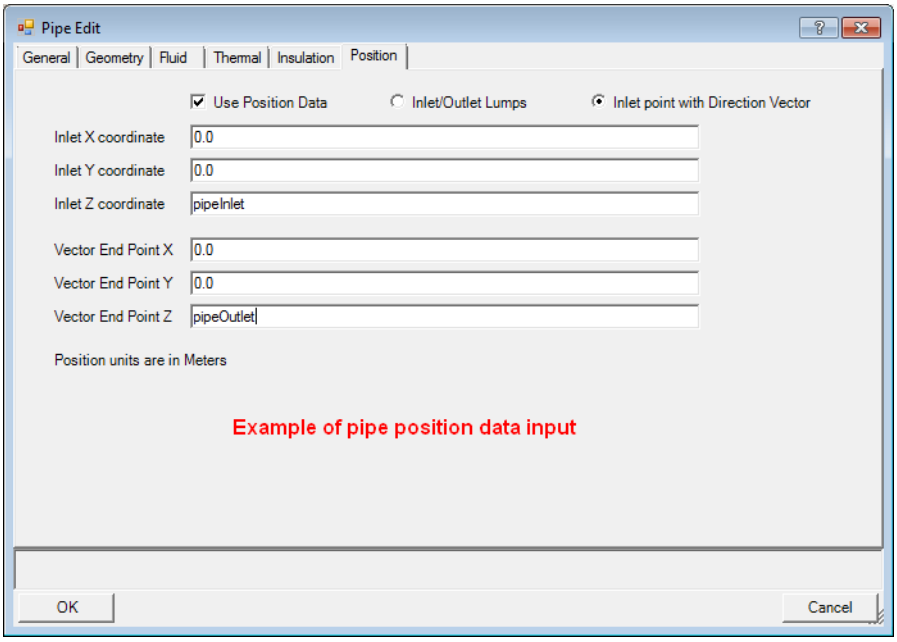
Insulation Definition - This portion of the form defines the various insulation layers. Layer 1 is located next to the pipe wall while subsequent layers build outward.

> *Node type* - Use the pull down list to select the node type for each layer, either arithmetic (no mass) or diffusion (with mass).
>
> *Material* - User the pull down list to associated material properties to each layer. If a material needs to be added, return to the thermal tab and add a material to the database by placing the curser in the material data field, click the middle mouse button and select Add Material. The dialog for adding a material to the database will appear. For more information on defining materials in the database see “Thermophysical Property Manager” on page 5-142.
>
> *Thickness* - Input the thickness of each layer. The input can be numeric, a register, or an expression.

* With the pipe highlighted, right-click the mouse and choose Insulation Visibility Toggle.

Pipe Position. Sinaps allows relative lump positioning for a pipe, similar to methods used for the positioning of fluid lumps, using the CX, CY, and CZ parameters defined within a three axis coordinate system. This portion of the Pipe edit form allows the user to locate the inlet of a pipe and orient the pipe in 3D space. There are two methods provided to position a pipe, *both assume that the pipe is oriented in a straight line*. Once the position data has been defined, Sinaps will apply position data to all lumps in the pipe.

> Caution: As stated, using position data in the Pipe form assumes a straight pipe between the inlet and the outlet positions. If this is not the case, lump position data may be used to define a pipe with bends, however be aware that using the lump position data conflicts with the Pipe form position data and subsequent editing of the pipe form will result in the warning that the pipe has been customized and the editing of the pipe at the pipe level becomes locked. The preferred method would be to break the pipe into shorter straight segments and make sure Automatic Duct Generation (see "Automatic Duct Generation" on page 7-6) is turned enabled, allowing Sinaps to "connect" the pipes into a duct.



Use Position Data - This field allows the user to enable/disable pipe positioning calculations. When enabled, Sinaps calculates the location of each lump in the pipe based on the position data provided by the user. Disabling this feature will suppress these calculations, hence not send fluid lump CX, CY, and CZ inputs to the processor. To allow the calculation of body forces on a lump, one or more corresponding gravitational accelerations must be defined in Global Control Data, see "Global Control Data" on page 9-19. Note that by default, all gravitational accelerations are defaulted to zero and although position data will be calculated, if no accelerations are defined, the gravitational effects will not be modeled.

Pipe Positioning Method - Two methods (described below) are provided for locating a pipe in 3D space: Inlet/Outlet Lumps (heritage method), and Inlet Point with Direction Vector (preferred method). Depending on which option is selected, the input fields on the edit form will change. For both methods, the X, Y, and Z values must be units of length based on the unit system the model is built in, meters for SI or feet for ENG. Inputs can be numeric, a register, or an expression.

Inlet/Outlet Lumps - This method allows the user to specify the x, y, z location of the inlet lump and the outlet lump. Sinaps will evenly located the remaining interior lumps along the length of the segment defined by these two points. Note, the segment length is used in the calculations, not the pipe length, therefore, if the pipe length changes, *the position data will not be updated*. This is a heritage method used by models built prior to version 5.6 but is still be useful for modeling certain types of pipes such as a helical coil. To have lump position data based on the pipe length, use the direction vector method.

Inlet X/Y/Z Coordinate - These are the X, Y, and Z coordinates for the *first lump* of the pipe.

Outlet X/Y/Z Coordinate - These are the X, Y, and Z coordinates for the *last lump* of the pipe.

Inlet Point with Direction Vector - This method allows the user to specify the x, y, z location for the inlet of a pipe, which may be the inlet of a path or a lump, and a second point (typically the outlet of the pipe) which allows Sinaps to calculate a direction vector. Using this method, Sinaps uses the inlet point to be the inlet (based on *original* flow direction of the pipe, indicated by increasing lump IDs along the pipe) and then orients the pipe using the second point. All intermediate lumps in the pipe are then evenly spaced along the direction vector using the actual length of the pipe defined in the Pipe Edit form. If the length of the pipe changes, the position data will automatically update to reflect the new locations. Since this methods determines lump locations based on actual pipe length, it is the preferred method. Note, this method was not available prior to version 5.6. All pre-5.6 models built with the Inlet/Outlet Lumps method will continue to use the heritage method until the user changes the selected method and appropriately modifies the input. This method is the default for all newly created pipes.

Inlet X/Y/Z Coordinate - These are the X, Y, and Z coordinates for *inlet of the pipe*, this will be the inlet of a path for a centered or downstream discretized pipe, or the first lump for an upstream pipe.

Vector End Point X/Y/Z - These are the X, Y, and Z coordinates for a second point on the vector (line) defined by the pipe, typically the outlet location of the pipe.

7.2.4 Editing Multiple Pipes

The Model Browser can be used to edit multiple pipes for a limited set of parameters (see“Browsing Pipes and Other Macros” on page 8-4). This is especially useful when several pipes need to be edited to change the lump or path type. This is the preferred method compared to editing the lumps and path individually or as a group outside of the pipe since such editing will result in the pipe becoming “customized” and limiting future editing of the pipes.

7.2.5 Moving Pipes

Pipes can be moved to a new location on the network diagram by placing the cursor over the frame of the pipe, click the left mouse button and hold while moving the pipe to a new location. Although pipes are a form of a collection (see “Collections” on page 4-15) they can be moved in to and out of other collections. When moved to a collection, the pipe will move to the same layer which contains the collection.

7.2.6 Connecting Elements Within Pipes

Pipes have a restriction that network objects within a pipe cannot be connected by the user to other objects within the pipe. This restriction is due to the fact that elements can be added/removed by the pipe form when the pipe resolution, wall radial resolution, or insulation is changed. Since Sinaps has now knowledge of how to handle user links in these situations, the restriction has been put in place. As a work around, the user should create a node/lump outside of the pipe and connect the pipe objects to it instead of each other.

7.2.7 Copying Pipes

Pipes can be copied by selecting the pipe and using network Edit menu or window short cut keys for copy and paste. After pasting the new pipe will be similar to the originally pipe with the exception of element numbering.

7.2.8 Connecting to Pipes

Sinaps provides connect ports ( ) to connect a pipe to the rest of the network. Inlet and outlet connect points are provided as discussed in “Creating a Pipe” on page 7-9. Additionally a multi-conductor port ( ) is provided allowing all thermal nodes in the pipe to be connected to a common node outside of the pipe, perhaps a convection environment. The multi-conductor port can only be used in associated with the multi-conductor object, “Multi-conductor Element” on page 7-27.

When two pipes are connected to the same lump, Sinaps will inspect key parameters and if appropriate, connect the two pipes into an extended duct macro, see “Automatic Duct Generation” on page 7-6. When a long continuous piping system is being created with changes in flow area, wall thickness, or insulation, or more resolution is required in specific parts of a pipe, each section should be created as a separate pipe and then the pipes should be connected. Care should be taken when defining diameters, flow areas, etc. to ensure the aggregator will connect the pipes appropriately. The network input check Show Created Ducts can be used as visual verification the pipes have been connected properly. If pipes of different flow areas are to be connected for a continuous duct, a path between the pipes will be required and should be defined an expander/reducer with unique inlet and outlet flow areas (AFI/AFJ) defined.

7.2.9 Pipe Orientation

Pipes can be oriented in several ways. By default, pipes will be created horizontally on the network with the thermal nodes below the fluid lumps. Sinaps allows pipes to be oriented horizontally or vertically, reversed, rotated 180 degrees, mirrored, or renumbered by selecting the pipe and then using the object edit menu which is accessible by right-clicking while hovering over the Pipe frame (edge of green rectangle).

Horizontal/Vertical Toggle - This option changes the display of elements within the pipe from horizontal to vertical, or the reverse.

Reverse - This option reverses the positive flow direction in the pipe without making any changes in the network layout.

Rotate 180 - Pipes can be rotated 180 degrees using the Rotate 180 option, which maintains the flow of the pipe in the forward direction, but allows the user to flip the layout of the objects. Instead of the default left to right numbering and flow, the numbering and the flow will go from right to left (or bottom to top if vertical).

Axis Mirror - This option changes the positioning of the thermal nodes relative to the fluid lumps for a pipe with a wall. By default, the lumps are on top of the nodes for a horizontal pipe, and they are on the left for a vertical pipe.

7.2.10 Selecting Network Elements in a Pipe

Elements within a pipe can be selected individually or in groups for editing, moving, etc. To assist in group selection, Sinaps provides a means to select all elements of a certain type in a pipe. Place the cursor over the frame of the pipe and click the right mouse button to access the object edit menu. The action list includes a Select option which expands to allow selection of all lumps, nodes, paths, conductors or ties.

7.2.11 Relabeling a Pipe

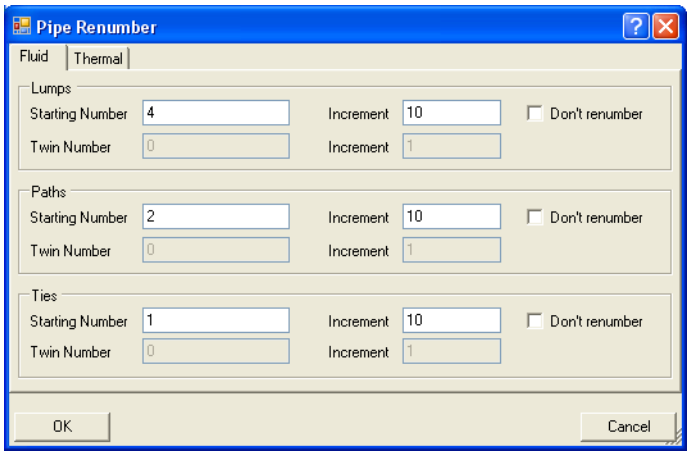
The default label for a Pipe is in the form PIPE-[Submodel Name].[pipe number]. Although visible in both expanded and collapsed mode, it may reside behind network elements when the pipe is expanded. The label can changed by clicking the label in the network to activate the label edit window, or by highlighting the pipe and then using the Object Properties panel to change the label, see“Network Object Label” on page 4-9.

7.2.12 Renumbering a Pipe

The Renumber option provides a means to renumber the objects in the pipe using the form shown in Figure 7-7. This renumber form allows the user to specify starting numbers for each object type along with increments. The Don’t Renumber check box allows the user to exclude certain object types from the renumbering process. The fields to renumber twins will activate if the pipe uses twins.

If a pipe is renumbered, it is up to the user to check and modify any user logic which may refer to the pipe elements.

FIGURE 7-7. Pipe Renumber Form



7.2.13 Customizing a Pipe

Once a pipe has been created, it may be customized by editing the individual elements. However, if a parameter is edited at the individual element which is normally controlled by the pipe form (e.g. path length, path wall roughness, lump or node volume, dup factors, etc.), the pipe will be flagged as being customized and further editing of the pipe through the pipe edit form is prohibited. Table 3 summarizes which parameters are controlled by the pipe edit form and when edited, will result in customization flags being set.

If elements in a pipe are customized, the pipe comment field on the general tab is a useful tool for documenting all customizations for future reference.

> Caution: Avoid customizing pipe elements outside of the pipe edit form. Once customized, pipe-level edits will become blocked preventing the ability to edit pipe-level parameters such as pipe length, diameters, resolution and element type, just to name a few. Instead of customizing a pipe, it is recommended that multiple pipes be created to model any axial variations such as changes in diameter, flow area, wall thickness or material, directional changes, or insulation properties. To maintain a continuous flow passage with multiple pipes, expanders/contractors should be modeled between pipes with dissimilar flow areas and automatic duct generation should be enabled.

Once an element has been “customized” outside of the pipe form, the pipe edit form can no longer be used to modify pipe level parameters. If used, clicking “OK” in the pipe form will overwrite all customizations of pipe level parameters. In such a situation, clicking the OK button in the pipe form will remove all customizations and the elements will be reset based on the contents of the pipe edit form. When customized, the user can inspect the fluid and thermal tabs of the pipe form to see which type of element has been customized. When a pipe is “reset” any parameter which may have been edited and is *not* controlled by the pipe edit form such as temperatures, k-factors etc., will not be reset.

Path inlet (AFI) and outlet (AFJ) flow areas are controlled by the pipe edit form. If an *individual* path must be customized for AFI and AFJ this can be done however the Variable Flow Area box in the pipe edit form must first be checked. If unchecked, Sinaps will not write path level AFI/AFJ values to the input (.sin) file. To customize a single path AFI/AFJ, at the pipe level check the Variable Flow Area box and define both the AFI and AFJ fields as the pipe area. After

saving the pipe form, edit the individual path to customized and define the AFI/AFJ. Be aware, that this customization will lock the pipe form from future editing.

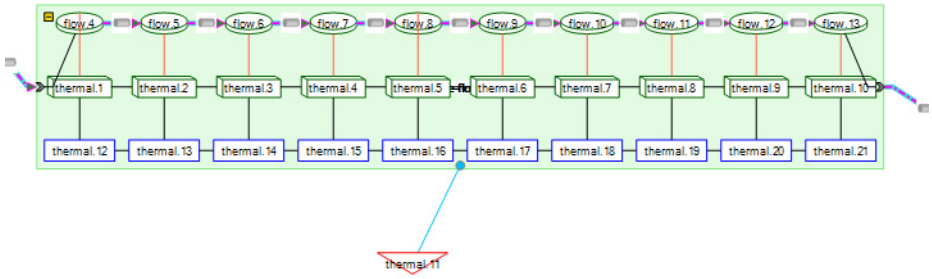
TABLE 3. Network Element Parameters Controlled by the Pipe Form

Element Type	Parameter	Exceptions
Lump	Lump type	
	Volume, VOL Position data, CX, CY, CZ	
Paths	Path type	Inlet and outlet flow area customization will be retained by checking the "Variable Flow Area" box.
	Flow area, AF	
	Inlet flow area, AFI	
	Outlet flow area, AFJ	
	Length, TLEN	
	Cross section	
	Hydraulic diameter, DH	
	Wall roughness fractions, WRF	
Ties	none	
Nodes	Node type	If pipe has a user-defined wall definition, the volume or capacitance customization will be retained and not be reset by the pipe form.
	Material	
	Volume/capacitance, VOL/C	
	Source, Q	
Conductors	Material	If pipe has a user-defined wall definition, conductance or area/length customization are allowed and will not be reset by the pipe form.
	Conductance or Area/Length, G	

7.3 Multi-conductor Element

The multi-conductor network element is designed specifically to connect the outermost nodes of a pipe to any single node in the network, possibly for environmental connections. When a pipe is created, a blue connect port ( ) will be located on the side of the pipe frame. This port can only be used for attaching multi-conductor elements to the pipe. Sinaps recognizes the purpose of the port and will create a multi-conductor when a link is drawn between the port and any thermal node. The resulting link (which represents many conductors) will also appear blue on the diagram. Only one node can be connected to a pipe using a multi-conductor element, however several multi-conductor elements can be attached to a single port.

An example of a multi-conductor is shown below. In this example, ten conductors are created between the boundary node thermal.11 and the outermost nodes 12-21 in the pipe collection. If the pipe has been defined with insulation, the connection will be made to outermost insulation nodes.



Multi-conductors can be thought of as a conductor generation tool. Although appearing to be a single link on the network, it will create multiple conductors, one for each outermost node along the length of the pipe.

> Warning: The numbering of generated conductors is controlled by Sinaps and cannot be modified by the user. In fact, ID numbers are not assigned to the conductors until the model is run and a SINDA/FLUINT input file is generated. Consequently, the numbering of the conductors can change from run to run depending on the availability of conductor ID numbers when the model is launched. It is recommended that generated conductors not be referenced in logic blocks since the IDs are variable. Instead, network element logic should be used.

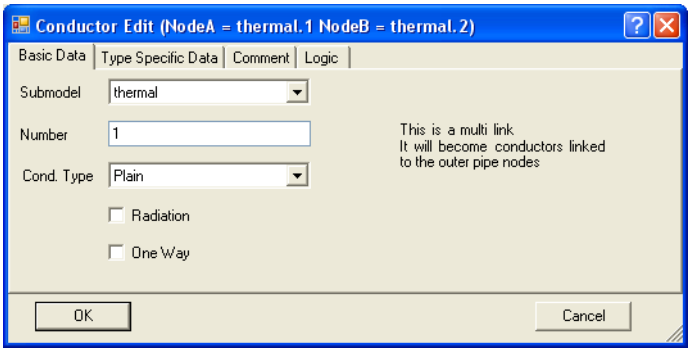
7.3.1 Creating a Multi-conductor

To created a multi-conductor, draw a link between the conductor port on a pipe to a thermal node on the network. By default the conductor will be assigned a conductance value based on current model defaults. By default the multi-conductor applies the conductance an area-based heat transfer coefficient h (units of W/m^2-K or BTU/hr-ft^2-R). The coefficient will be multiplied by the outer area (A) of the pipe or insulation ("h*A") and then divided equally across the outer wall nodes to calculate a conductance. Alternately, the user can specify a material and thickness (t) to apply a "K*A/t" type conductance calculation.

Note that when connecting to a "user defined" pipe, or a pipe with a wall defined as mass/length or cross sectional area, a outer perimeter field will activated in the pipe form. This field and the pipe length are used by Sinaps to calculate the surface area (A) of the pipe for the multi-conductor attachment.

7.3.2 Editing a Multi-conductor

To edit a multi-conductor element, double click on the link in the network to access the edit form. Alternatively, the multi-conductor can be located in the model browser and the model browser Edit option can be used to bring up the edit form. The edit form is shown below is a tabbed input form. .



Multi-Conductor Basic Data Tab. This tab allows the user to input basic conductor data.

Submodel- The submodel in which the conductors will reside. By default, this will be the current submodel. This field can be used to change the submodel.

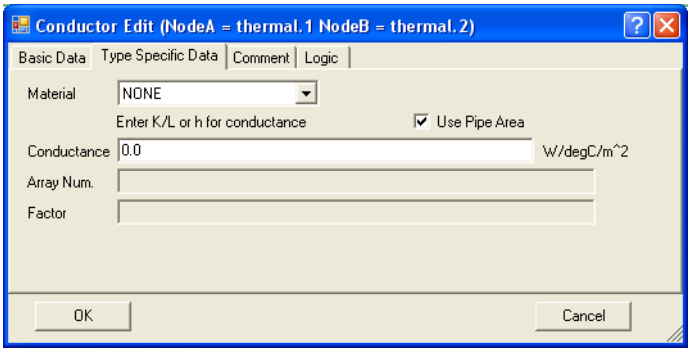
Number - The identification number (integer) for the multi-conductor object. This is used for identifying the multi-conductor object in the model browser and for identifying the generated conductors in the .sin file. This field does not control the numbering of the conductors to be generated.

Cond Type - This field is used to identify the type of conductor to be created: linear, temperature-varying (SIV) or other (DIV, SIM, BIV etc.) If temperature-varying is selected, an array number and factor will need to be defined on the Type Specific Data tab.

Radiation - Check this box if the conductor is to be treated as a radiation conductor. By default radiation conductors are multiplied by the global control constant SIGMA. SIGMA defaults to 1.0. For radiation conductors either set SIGMA in global control data to the Stefan-Boltzmann constant or include it in the conductor definitions. Whichever method is implemented, the user must be consistent for all radiation conductors in the model.

One-way - If check the conductor will be a one-way conductor. When designated a one-way conductor an arrow will appear on the link in the diagram. The node on the arrow side of the link will "see" the conductor and any subsequent heat gain or loss. The node at the base of the arrow will not "see" the conductor and will not see any heat flow through the conductor.

Multi-Conductor Type Specific Data. This tab allows input of type specific data. The contents of the form will vary depending on the type of conductor selected on the basic data tab.



Material - Conductors can be defined using the thermophysical propertied data base as an alternative to inputting the full conductance directly. If so desired, use this field to select the material to used with the conductor. When a material is selected, the conductivity from the database will be used. In this case the conductors input should not include a conductivity.

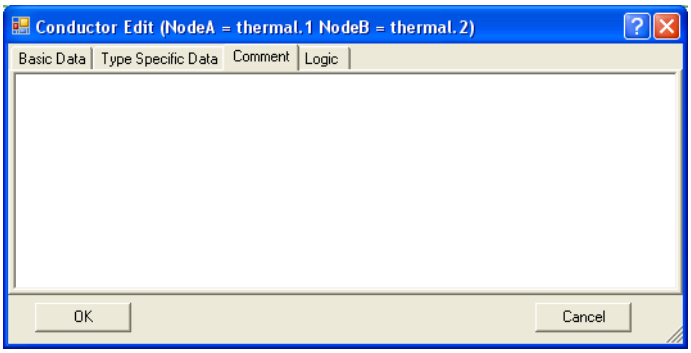
User Pipe Area - When checked (default) the surface area of heat flow will be determined by the parameters defined in the pipe form (length, outer diameters or outer perimeter). When not checked, the area should be included in the conductance data input field on this form.

Conductance - Applicable when no material is selected. This is the *total* conductance term to be divided evenly across all generated conductors. The input will vary depending on whether the Use Pipe Area box is checked or not checked. The conductor field is designed to accept inputs as numeric values, registers, or expressions. For more information on expressions see “Parameterizing a Model” on page 6-1.

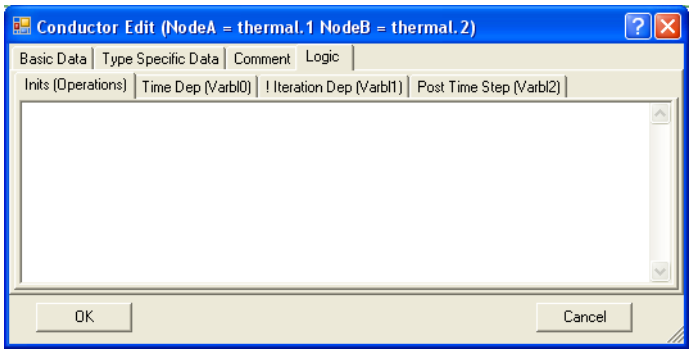
Array Number- If the conductor type is selected as Temperature Varying (SIV), the array number field should reference the user-assigned number for the array containing the temperature vs. conductivity data. The user must have already created the array through the Array Data block accessible through the Advanced tab of the Case Manager.

Factor - Applicable only for temperature varying conductors. The factor field can be used to define a multiplier for the conductivity such as area/length.

Multi-Conductor Comment Tab. The comment tab can be used for documentation purposes.



Multi-Conductor Logic Tab. This field can be used to define network element level logic to be *applied to every conductor* generated. Note that when using network logic with multi-conductors, unlike most element level logic, the specific conductors cannot be referred to when creating logic since the conductor IDs are not know, however indirect operators such as "#this", "#nodeB", and "G#this" can be used. For details of using network element logic and indirect referencing, see "Network Element Logic" on page 5-137. The "#nodeA" side of each generated conductor will be the single node (outside the pipe component), whereas the "#nodeB" side of each generated conductor will be the outer-most nodes on the pipe wall or insulation.



7.3.3 Post-processing a Multi-conductor

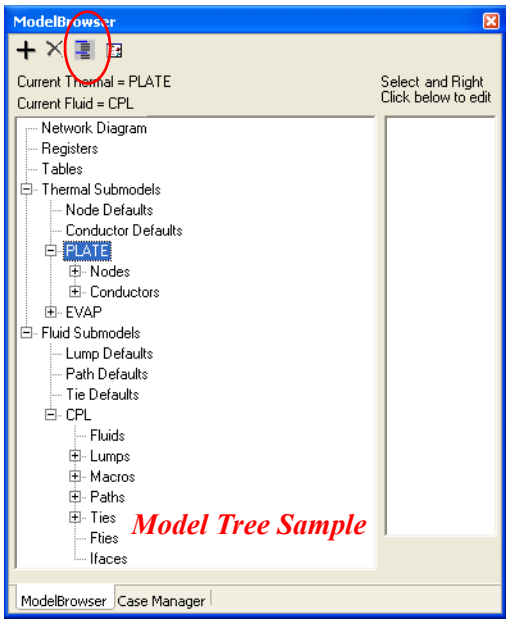
Multi-conductors can be post-processed as text or an XY plot once a model has been run. To post processes a multi-conductor, first use the case manager to select the conductor parameter for post processing (see "Selecting Data to Post-process" on page 10-13, then select the link representing the conductors. Next either plot the results or use the network post-processing option Text to bring up the results in an Excel table. In either case, all conductors which have been created by the multi-conductor will appear in the post-processed data.

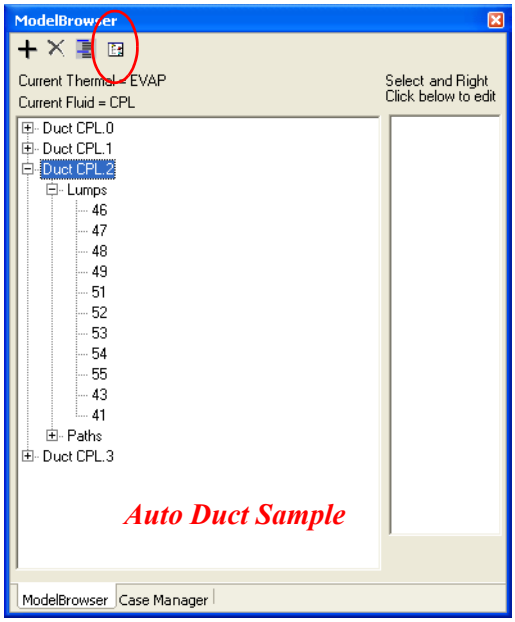
When a post-processing tool (gauge or meter) is attached to a multi-conductor, the function field in the tool will activate. This allows post-processing for the sum, average, minimum, etc. of all the conductors represented by the multi-conductor.

CHAPTER 8

The Model Browser

The Model Browser is a task panel providing access to and easy navigation of the network elements.* It also provides access to the network diagram, registers, tables, object defaults and fluid definitions. Being a task panel, the Model Browser can be docked to the frame of the work space, undocked, or it can be tabbed with other task panels. If the Model Browser has been closed, it can be retrieved using the pull-down menu option View>Model Browser. The model has the ability to show either the model tree or a tree of automatically generated ducts. Icons at the top of the window control the view.





* While Thermal Desktop® users will be very familiar with the analog of the Sinaps Model Browser, Sinaps*Plus*® users should take a little extra time to familiarize themselves with this important new feature.

8.1 The Model Tree

The Model Tree displays all network elements (objects and links) in a tree-type structure, arranged by submodel. Clicking the [+] to the left of any submodel will expand the submodel-level tree. Within each submodel, elements are arranged by type (e.g., Nodes, Conductors, Lumps). When an element type is clicked in the tree in the left pane of the Model Browser, the right pane will populate with the integer IDs for the selected type of element in that submodel. This provides an alternative method for navigating a model and locating modeling elements by type and ID.

The model tree also provides a means for setting default data field values for nodes, conductors, lumps, paths, and ties, and for access to define fluid properties for each fluid submodel. Additionally it provides access for editing, deleting, and locating elements. When a network element is selected in the pane on the right of the model tree, clicking the right mouse button will provide access to a pop-up menu (Edit, Color in Network, Reset Colors). These options are explained below.

8.1.1 Using the Submodel Tree

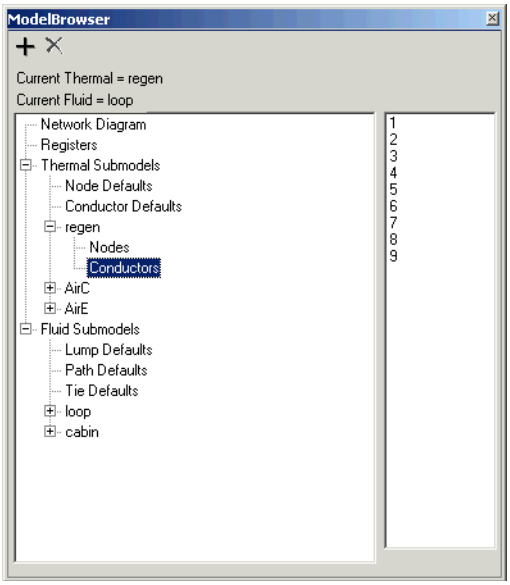
Submodels can be added, deleted, renamed, exported and imported through the Model Browser. For information on adding/deleting submodels see “Creating Submodels” on page 5-7. Submodels can also be imported or exported renamed through the model browser by selecting a submodel name and then using the main menu option Model>Submodel>Import/Export. For more information on submodel import/export please see “Submodel Import and Export” on page 11-5.

Submodel Rename: The user may rename a submodel by highlighting the submodel name in the model browser and then clicking the main menu option Model>Submodel>Rename. After providing a new name the Find and Replace dialog will appear prompting to replace name. with newname. in the model. This dialog is used to search expressions and logic to update all occurrences of the submodel name, typically used in processor variable references such as widgit.T10 (the temperature of node 10 in submodel “widget”). Note that other user defined searches may be required to change all occurrences such as changing ‘name’ to ‘newname’ (bracketed with single quotes) to change the submodel name in subroutine call statements within user logic. If the model contains a user define build statement or calls to DEFMOD (setting default model in logic blocks) they too may need to be changed manually.

Editing network elements: Using the model tree, the user can edit network elements. For example in Figure 8-1, the user can navigate to submodel REGEN, select Conductors and a list of all conductors in the submodel will appear in the right pane of the browser. After selecting the conductor of interest, the object edit window can be accessed by clicking the right mouse button to access the pop-up menu and then selecting Edit. Multiple conductors may be edited by holding down the shift key and selecting two or more conductors.

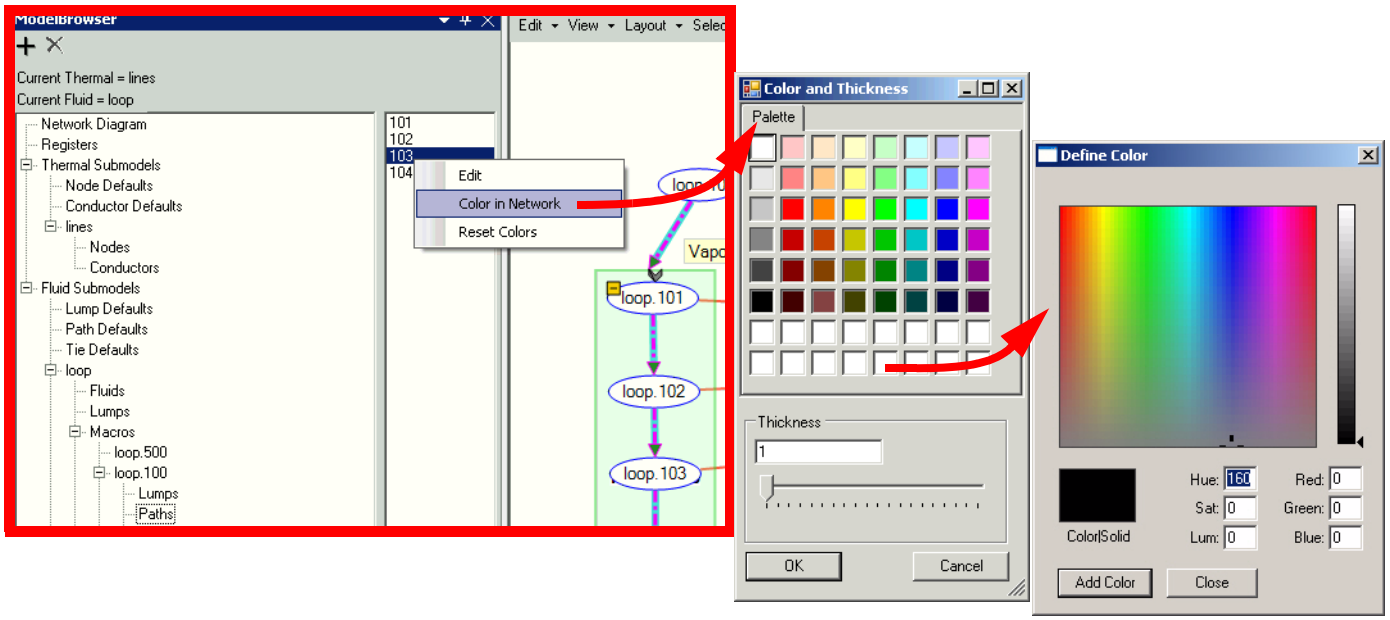
Alternatively, any network element can be edited in the Object Properties window. When an element ID is selected in the Model Browser, the Object Properties window will automatically update to show the information for that element, providing access to edit any of the data fields.

FIGURE 8-1. The Model Browser



Highlighting elements in the network: Right clicking on the ID of any single object shown as Figure 8-2 will display the object menu. To color the element, right click on the ID and the select Color in Network and select a color or right click in one of the white color cells in the bottom two rows to access the color pallet and create a custom color. To thicken a network link (path, conductor, etc.) move the Thickness slider bar or input a value in the data field. Multiple elements can be highlighted at once using this method. When the highlighting operations have been completed, use Reset Colors to return the network to the original state.

FIGURE 8-2. Highlighting Network Elements with Model Browser



Deleting network elements: Locate the element ID to be deleted in the model tree, select it and then hit the delete key. The element will be removed from the network diagram and the model tree. Please note that deleting a network object will also delete in any links between it and the network.

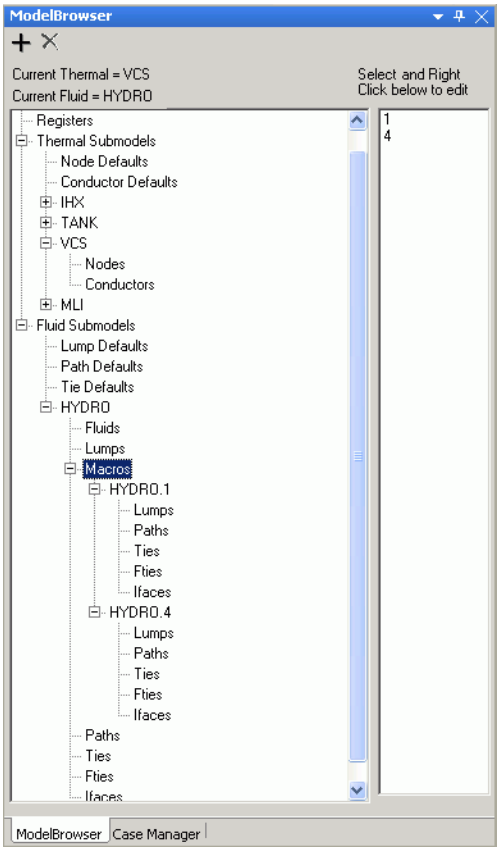
8.1.2 Browsing Pipes and Other Macros

SINDA/FLUINT macros (LINE, HX, and CAPPMP) are listed in the Model Browser under the submodel in which they have been created. Since Sinaps Pipes contain SINDA/FLUINT LINE and HX macros, they will appear as fluid macros *only* under the fluid submodel portion of the Model Browser and contain only the fluid elements. Since macros are fluid only objects, if an HX macro has been created, the nodes and conductors will *not* appear under the macro option in the browser. Rather, they will appear under nodes and conductors in the thermal model portion of the browser.

Each macro can be expanded in the left portion of the browser to display a listing of network elements contained in the macro. Clicking on any portion of the macros, such as paths, will result in a listing of the individual paths in the right pane of the browser window. Right-clicking on any macro element ID in the right pane (Figure 8-3), will bring up the object menu allowing the object to be edited (in the appropriate edit form) or highlighted in the network for easy identification as explained above.

Likewise clicking on the `Macros` entry in the left pane will result in a listing of all macros in the right pane. If one or more macro IDs are selected in the right pane, a select group of parameters can be edited in the object properties window. This is useful for changing lump or path types for several pipes at once.

FIGURE 8-3. Browsing Pipes (Line/HX Macros)

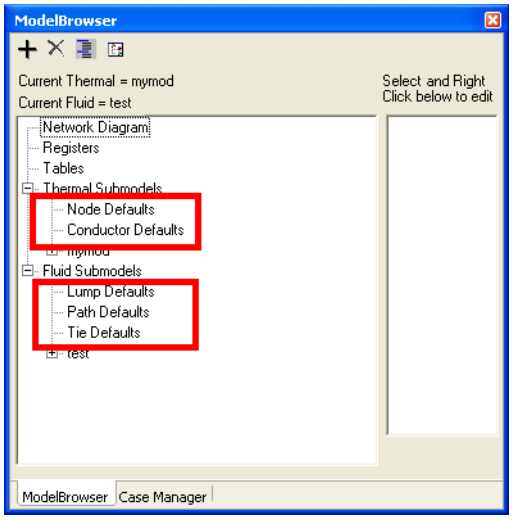


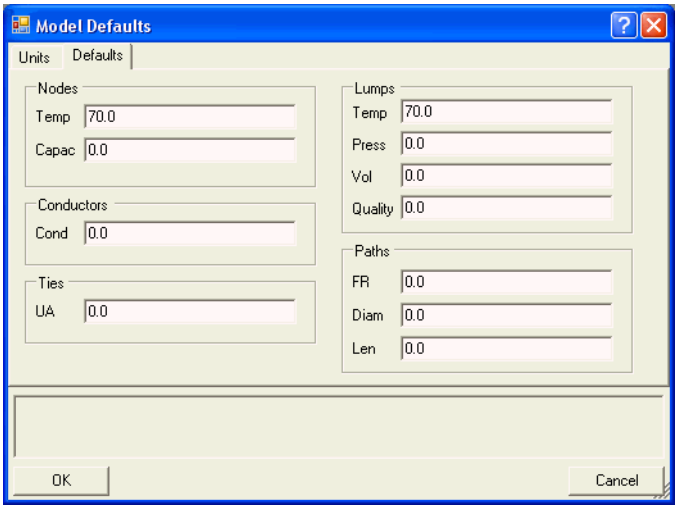
8.1.3 Setting Network Element Defaults

The Model Browser provides a means for the user to set defaults for the various network elements, thus reducing the amount of editing that is otherwise required after creating objects and links. Once the defaults for a network element have been defined, Sinaps will use those values for all subsequent elements created in the network diagram. Defaults can be modified at any time during model construction. Defaults can be reset to Sinaps default by right clicking on a network element and selecting Default>Reset default.

Caution should be used when setting defaults. Defaults apply all future network elements; setting defaults has no impact on existing network elements. Note that if setting defaults for a type of network element, the default will apply to *all subtypes* of that element. For example if setting defaults for a path, the new defaults will apply to all path types which are many. For example, if you set a high K-factor on a loss to represent a valve and then make that path your default, all subsequently created paths including tubes and stubes will have that high K-factor. In this situation after creating the new loss paths to represent valves based on the default, the *path defaults should be reset to avoid inadvertantly setting unwanted and perhaps unapplicble parameters on future paths*.

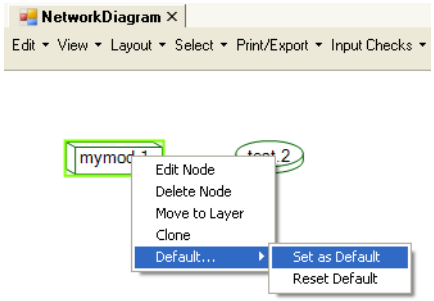
How to set defaults: When a new model is started, Sinaps will automatically prompt the user to set initial defaults for network elements (See “Setting Units for the Model” on page 4.) During the course of creating a model, the defaults can be modified by accessing the defaults dialog through the menu system (Model>Defaults & Units) or by using the Model Browser option below. Alternatively any network element default can be set to be the same as an existing customized element by right clicking on the element and selecting the Default>Set as Default option. Note this option sets defaults for all parameters, some of which may not be accessible through the Model bBrowser or the Model Defaults form. When using the Set as Default option, Sinaps will *not* include fluid element twinning in the defaults. Twinning must be applied manually to fluid elements or at the Pipe level.





To set defaults using the Model Browser tree, first select the *type* of defaults. For example, select Node Defaults in the tree to set the node defaults. This action will cause “Node Defaults” to appear highlighted in the tree, and the Object Properties window to automatically appear and to display the available defaults for that element type. Input the desired values in the various data fields of the Object Properties window.

While working in a model, defaults for a type of network element (node, lump, path, etc.) can be set based on an existing object or link on the network by right clicking on the network element and selecting Default>Set as default. This method for setting default sets all parameters associated with the device except twins and dup factors. Many of these parameters may not be available through the Model Defaults form or the Model Browser. All subsequent elements of this type will be identical to the object used to set the defaults. Since links (paths, conductors, etc.) cannot be copied, this is useful in lieu of copy/paste.



Resetting Defaults: Once defaults have been set through the default form, the model browser, or the right click option, they can be reset to Sinaps default values by using the right click Default>Reset Defaults option on any type of network element. Note this will only reset the defaults for that particular element type (lump, node, path, etc.) not for all network elements.

8.1.4 Specifying Fluids

The Model Browser provides the primary means for defining fluid properties. When a fluid submodel is created, the working fluid (or fluid mixture) must be defined. In the model tree, navigate to the fluid submodel and expand the submodel (click the [+] to the left of the submodel name). The first item in the submodel tree will be Fluids. When the fluids option in selected, the right pane of the Model Browser window will display the fluid(s) for that particular submodel. To edit the fluid or add additional fluids for a mixture, select the fluid in the right pane, right click the mouse, and select Edit Fluids. The fluids editing window will appear. Options are available for editing, adding or deleting a fluid. For detailed information on defining fluids please see “Working Fluids” on page 5-19.

8.1.5 Network diagram

The first entry in the Model Browser tree provides access to the Network Diagram. (This action is equivalent to selecting View>Network from the main workspace pull-down menu.) The Network Diagram is a sketch pad containing the user created resister-capacitance (R-C) schematic of the model, similar to an electrical schematic. The diagram shows capacitances in terms of thermal capacitances and fluid mass (nodes and lumps) connected by resisters, paths of heat, mass or energy flow. For detailed information on the Network Diagram see “Working with Networks” on page 4-1.

8.1.6 Registers

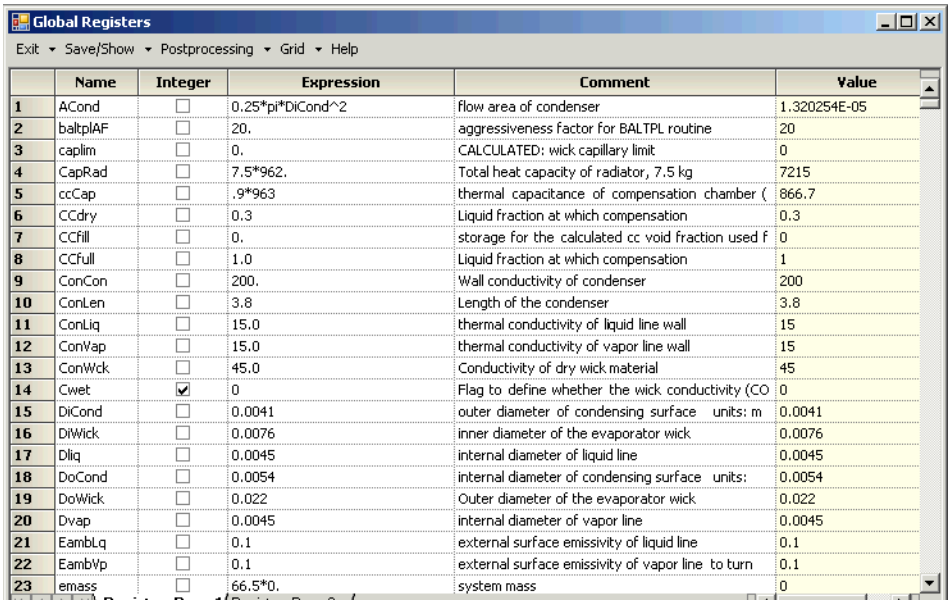
The register spreadsheet shown in Figure 8-4 is accessible through the Model Browser. This spreadsheet contains user defined variables called ***registers*** which can be defined as a unique number or an algebraic expression based on other registers and processor variables. Processor variables are output parameters such as temperature or flowrate for which a value is not defined prior to executing the model. Registers can be used stand alone or in algebraic expressions in any data entry field to aid in rapid model changes, parametric studies, and model documentation. The user may define up to

15000 named registers. The register spreadsheet features various built-in functions as well as built-in constants such as π (“pi”) as well as common unit conversion factors.

> Note: Registers in Sinaps are identical to registers in SINDA/FLUINT but they are **not** the same as ***symbols*** in Thermal Desktop. Thermal Desktop symbols can be imported as registers. However, Thermal Desktop array-based symbols and not support by Sinaps and will be ignored at import.

Registers can be used as design variables for the Solver or as random variables for Reliability Engineering, both part of the advanced design module of SINDA/FLUINT. Because registers help auto-document a model and are key to many powerful options, extensive use of them is strongly encouraged. For detailed information on creating and using registers see “Parameterizing a Model” on page 6-1.

FIGURE 8-4. Register Spreadsheet

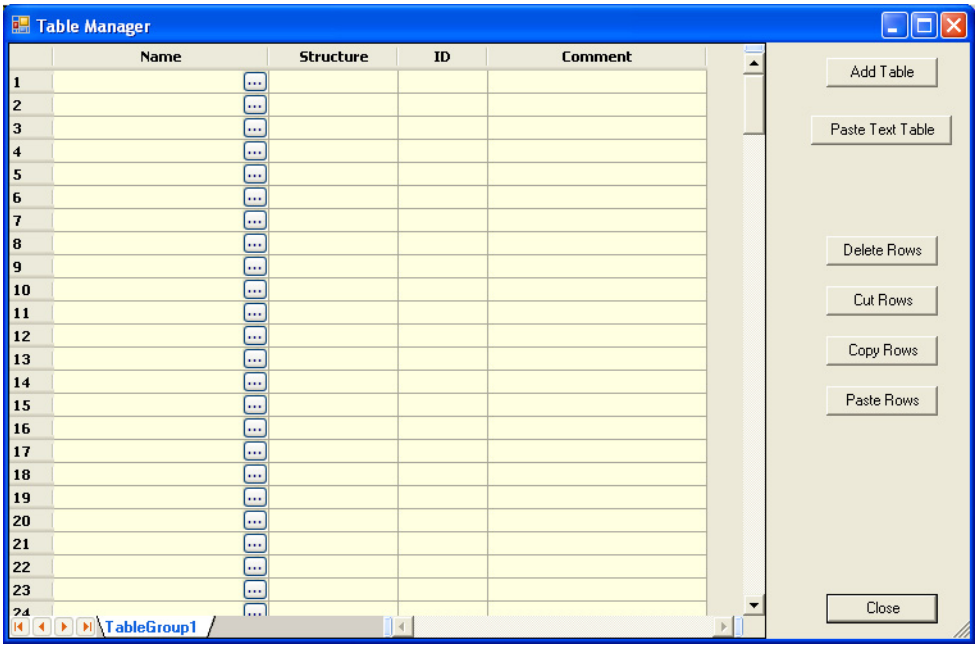


	Name	Integer	Expression	Comment	Value
1	ACond	☐	0.25*pi*DiCond^2	flow area of condenser	1.320254E-05
2	baltplAF	☐	20.	aggressiveness factor for BALTPL routine	20
3	caplim	☐	0.	CALCULATED: wick capillary limit	0
4	CapRad	☐	7.5*962.	Total heat capacity of radiator, 7.5 kg	7215
5	ccCap	☐	.9*963	thermal capacitance of compensation chamber (	866.7
6	CCdry	☐	0.3	Liquid fraction at which compensation	0.3
7	CCfill	☐	0.	storage for the calculated cc void fraction used f	0
8	CCfull	☐	1.0	Liquid fraction at which compensation	1
9	ConCon	☐	200.	Wall conductivity of condenser	200
10	ConLen	☐	3.8	Length of the condenser	3.8
11	ConLiq	☐	15.0	thermal conductivity of liquid line wall	15
12	ConVap	☐	15.0	thermal conductivity of vapor line wall	15
13	ConWck	☐	45.0	Conductivity of dry wick material	45
14	Cwet	☑	0	Flag to define whether the wick conductivity (CO	0
15	DiCond	☐	0.0041	outer diameter of condensing surface units: m	0.0041
16	DiWick	☐	0.0076	inner diameter of the evaporator wick	0.0076
17	Dliq	☐	0.0045	internal diameter of liquid line	0.0045
18	DoCond	☐	0.0054	internal diameter of condensing surface units:	0.0054
19	DoWick	☐	0.022	Outer diameter of the evaporator wick	0.022
20	Dvap	☐	0.0045	internal diameter of vapor line	0.0045
21	EambLq	☐	0.1	external surface emissivity of liquid line	0.1
22	EambVp	☐	0.1	external surface emissivity of vapor line to turn	0.1
23	emass	☐	66.5*0.	system mass	0

8.1.7 Tables

The Sinaps ***Table Manager*** is accessible through the Model Browser by clicking on Tables. The Table Manager, shown in Figure 8-5, provides a means to define SINDA/FLUINT arrays or any other type of columnated or tabulated data in a table format. Table contents can be referenced by network elements or user logic. For more information on tables, please see “User-defined Data Tables” on page 5-121.

FIGURE 8-5. Table Manager



8.2 *Automatic Duct Tree*

The model browser also provides access to a listing of all automatically generated ducts (see“Automatic Duct Generation” on page 7-6). This tree lists all automatically generated ducts in a tree structure to show which network elements have been aggregated into the automatic duct. Like the model tree, the duct tree has options to right click on any duct and color it in the network.

CHAPTER 9

Using the Case Manager

9.1 Managing Analysis Cases

The purpose of the ***Case Manager*** is to allow the user to set up multiple analyses (SINDA/FLUINT runs) based on a single master model.* For example a single model may contain both a steady state simulation and a transient simulation, with individual customizations for each ***case***. Both simulations use the same underlying model network, but the logic (and perhaps register values, etc.) might be different for each case. Alternatively, cases may build different submodels from the master model, such as stowed and deployed configurations of a radiator or antenna.

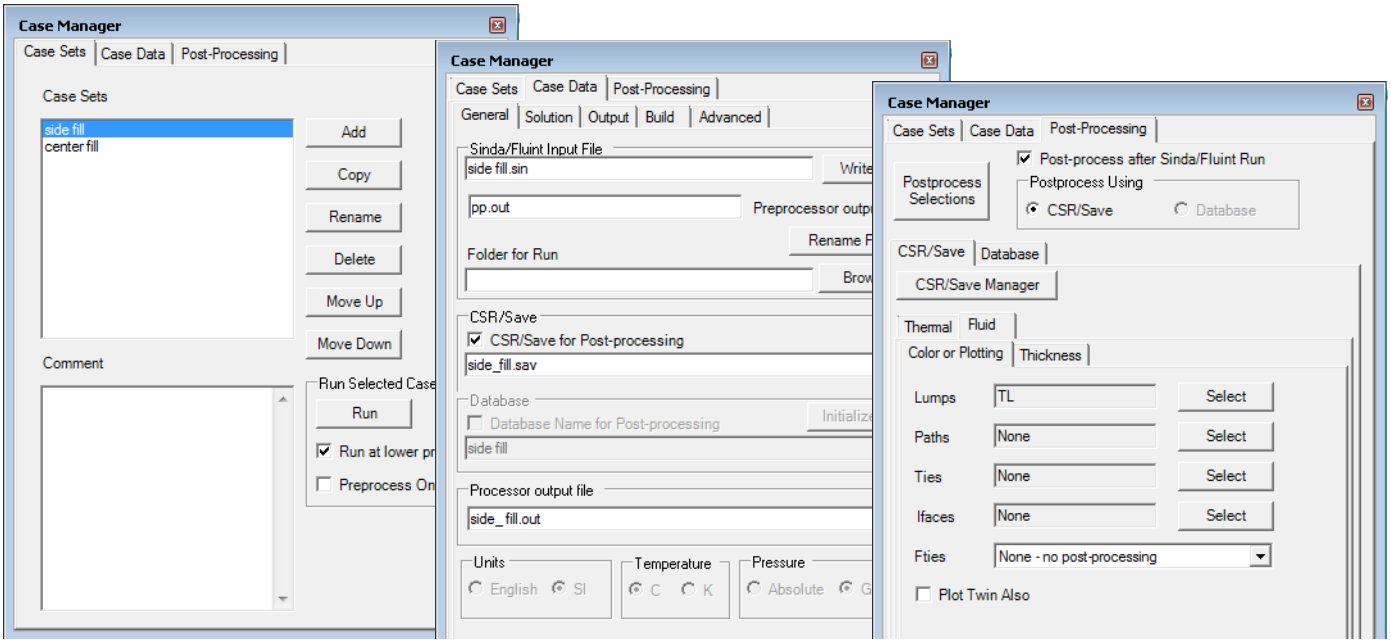
Once a case is defined, with the click of a single button the model file can be written, SINDA/FLUINT launched, and the results automatically post-processed on the network diagram.

The remainder of this chapter will provide usage instructions for the Case Manager task panel. If the panel has been closed, it can be reopened using the pull down menu View>Case Manager. Like other task panels, it can be docked, tabbed, or undocked from the frame of the application window.

The panel contains three tabbed forms as shown in Figure 9-1: Case Sets, Case Data, and Post-processing. The Case Sets tab is where cases can be added, deleted, renamed, and managed. For each case defined in the upper pane, the lower pane can be used to provide a description of the case. The Case Data tab (section 9.3) provides the ability to define output file names, define the type of solution, define which set of submodels contained in the master model are to be built, define control data, access and create user logic and more. The Post-processing tab (section "Case Manager Post-processing Tab" on page 10-10) allows the user to select post-processing parameters for the network and plotting.

* While Thermal Desktop® users will be very familiar with the analog of the Sinaps Case Manager, Sinaps*Plus*® users should take a little extra time to familiarize themselves with this important new feature.

FIGURE 9-1. Case Manager Task Panel



9.2 Defining Cases

The Case Sets tab is used to define and organize the cases for a master model. By default, the case manager contains a case called “base” which runs a steady state simulation with default settings. The user can modify this case or create new cases.

Cases can be created and deleted by the user. They can also be moved up or down in the listing, The order of cases determines launch order if multiple cases are run. The lower portion of the Case Sets tab contains a Comment window. This window may be used for providing a description of the case or any other information the user would like to associate with a particular case. Additional model documentation can be created through the main pull-down menu Model>Notes.

To Create a Case: New cases can be added by either creating a new case using the Add button or by copying an existing case by clicking to highlight a case and then clicking on the Copy button.

When a new case is ***added*** it will be an “empty” case running a default steady state simulation with no logic blocks and no subroutine data, (unless previous multi-case designations have been made on the Advanced tab, see “Case Data - Advanced Options” on page 9-13). The user will be prompted to provide a new case name when adding a case. The case name is limited to 72 alphanumeric characters, and spaces are allowed. This case name will be used for default naming of the sinda input and output files.

When a case is ***copied***, all customizations including the addition of user logic, control data, case level registers subroutines, etc. will be retained and propagated to the new case. Again the user is prompted to provide a new case name

which will be used for unique file naming. After a case has been copied, the cases are treated as unique and independent. Hence subsequent changes to the original case will *not* be applied to its copy. Provisions are available to make multi-case changes that are applicable to all cases, as described in “Case Data - Advanced Options” on page 9-13.

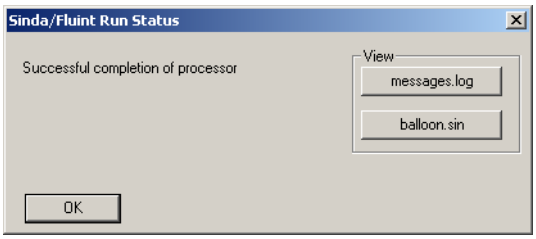
To Rename a Case: The user can rename a case by clicking the Rename button. When a case is renamed, all associated input and output files for that case are also renamed.

To Delete a Case: The user can delete a case by clicking on the Delete button. *Once a case deleted, this action cannot be undone other than exiting the model without saving.*

To Run a Case: To execute any case, simply highlight the case name and click the Run button. Sinaps will write out an input deck and launch SINDA/FLUINT automatically. By clicking the Run at lower priority check box below the Run button, the user can specify that the process run at lower priority. This option gives the other processes on the computer a higher priority by running the model in the background so that the user can continue working in other applications. The Preprocess Only check box provides the option to run just the preprocessor. This is useful for error checking a model before running. For more information on error and input checking see “Input Checking” on page 4-17.

By holding the shift key and selecting multiple cases, several cases can be run back-to-back. Each case will be run independently so if one case needs to start from the final state of a previous model, the necessary logic must be included to save the final processor data and read in that data for initial conditions in the subsequent case. For more information on “restart operations” please see section 4.7.3 of the SINDA/FLUINT User’s Manual.

After a model has been launched, a status window will appear on the screen showing the iteration count for a steady state case or the time for a transient case. When the execution is complete a window will appear stating the model has completed.



Generally, one of five messages is displayed:

- *Preprocessor failed - see preprocessor output file and messages.txt in your input file subdirectory* - This message indicates that SINDA/FLUINT was unable to begin execution of this model due to a formatting error, spelling mistake in logic, network construction violation, missing INCLUDE file, etc. Search the pp.out file for the words ERROR and CAUTION to determine the cause. The generated text input file (*case_name*.sin) is available for inspection as well.
- *Compile or link of astap.for failed - see messages.txt in your input file subdirectory* - This message indicates that the Fortran compiler has failed even though the model preprocessed successfully. This is usually caused by an unresolved external (e.g., calling a routine that wasn’t supplied or that was misnamed), a problem in calling arguments, logic violating column conventions, missing or unequal parentheses, DO loops not closed properly with ENDDO statements, etc. The *astap.for* file is the main program generated by the SINDA/FLUINT preprocessor, and it can be searched for the line number(s) reported in the *message.txt* file for hints.

- *Successful completion of processor* - This message indicates the model has run successfully with no errors or cautions. Nonetheless, the user should make sure that the results are reasonable.
- *The model has completed successfully. However there are cautions/errors the user should be aware of.* - This message indicates the run is complete but there are caution or error messages associated with the run that were generated by the SINDA/FLUINT processor. The user should check the *messages.log* file and review the messages. These message may include energy imbalances, convergence issues, etc. It is the user's responsibility to determine if the resulting output is valid. Additional information on energy imbalances and nonconvergence is available in the output file (*.out), with general guidance available in the SINDA/FLUINT User's Manual.
- *Processor failed - see output file and messages.txt in your input file subdirectory.* This message means there was a fatal error of some type preventing the model from completing. This type of error may include numerical problems encountered in user logic (e.g., overflows or divide-by-zero) which leave only a trace-back of routines (search *messages.txt*) called along with line numbers to search in the *astap.for* file.

  This may also be caused by a program abort when a severe error was encountered during processor execution. The user must look at the messages (*messages.log*) and determine the cause of fatal error. Note that the standard SINDA/FLUINT abort routine is called ABNORM, and that output calls are invoked after it has been triggered to try to provide clues as to the cause and the state of the network when the problem was encountered. Therefore, *search in the output file (case_name.*out*) above the last set of output* to find the message describing the abort cause.

9.3 Setting Case Data

Once a case name has been highlighted in the Case Sets tab, its run parameters and post-processing settings may be edited under the Case Data and Post-Processing tabs of the Case Manager.

The Case Data tab in the Case Manager allows the user to set necessary control parameters and execution logic for each case. This is where the model units are set, the type of solution is defined, build configuration is defined, desired output is identified, convergence criteria are specified, and any supporting user logic may be defined.

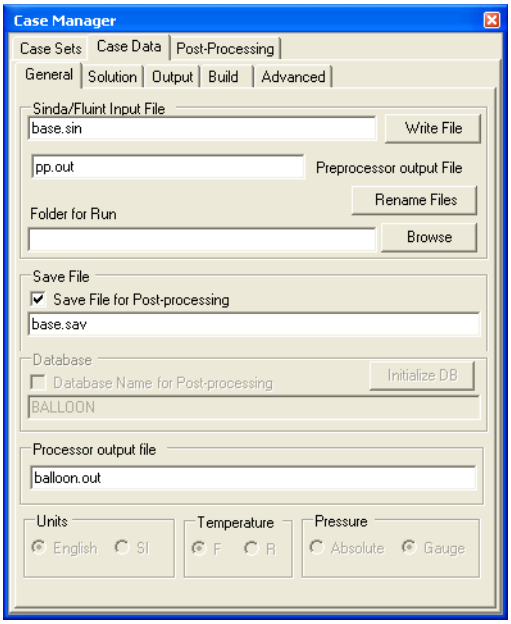
By default, the case data is unique to each case defined in the Case Manager. Some case data information may also be designated to apply to multiple cases (see "Case Data - Advanced Options", Section 9.3.5).

9.3.1 Case Data - General

The General tab provides a location for defining the name for the input file, preprocessor output file, processor output file, and save file or database (Figure 9-2). By default, the input file name is [case name].sin, the preprocessor output is *pp.out*, the processor output is [case name].out and the save file is [case name].sav. These names can be changed to be consistent throughout the case (replacing the case name as the default naming scheme) using the Rename Files button.

Model units are displayed on the General tab for reference only. For more information on units see "Setting Units for the Model" on page 5-4.

FIGURE 9-2. Case Data - General Tab Options



SINDA/FLUINT Input File: The user can write out the text-based input file ([case name].sin) by clicking the Write File button. The preprocessor output file contains information useful if the model fails with preprocessor errors. In such a case, open the preprocessor output file and search for the therm “error” to identify the cause of the failure.

Processor Output File: The processor output file is the text output created when a model is run. This file will contain important messages to the user such as cautions and warnings along with any text output requested by the user on the Output tab in the Case Manager

Run Folder: By default, the model is run in the same directory as the model file. However, the user can specify a different directory by defining a location in the Folder to Run data field. Clicking the Browse button will bring up a directory browser window to locate an existing directory (or create a new one using the Make New Folder option at the bottom of the browser form).

> Tip: Since several files are created when running a model, users may find it useful to create distinct subdirectories for each case.

CSR/Save: When a model is executed all generated binary data (processor results) are stored in either a CSR Folder, a SAVE file, or a database. The content of the processor results is controlled by the user via the Output tab in the Case Manager.

CSR/Save for Post-processing (check box) - This check box (checked by default) must be checked to enable writing the processor results to a binary CSR folder or SAVE file. When checked, the name of the CSR folder or save file is written to Options Data and a CALL SAVE is automatically added to the model logic.

CSR/Save (Name) - This field is the name of the CSR Folder or SAVE file. The name is automatically defined based on the case name and whether or not a CSR Folder or Save File is being used. By default Sinaps creates SAVE files. The user can optionally change a model default to a CSR Folder by selecting the menu option Model>User CSR Folder. The save file is a binary file which is used by Sinaps for post-processing the network diagram and by the EZXY Plotter for plotting data. The Database option (currently not active) is for future expansion.

Database - In addition to, or as an alternative to the CSR Folder and SAVE file, the user can opt to put all generated processor output in a database. This feature is currently deactivated but will be available in the future.

Processor Output File- This is the name of the text-based output file generated by the processor. By default the name is based on the case name, but the user may override this.

. *9.3.2 Case Data - Solution*

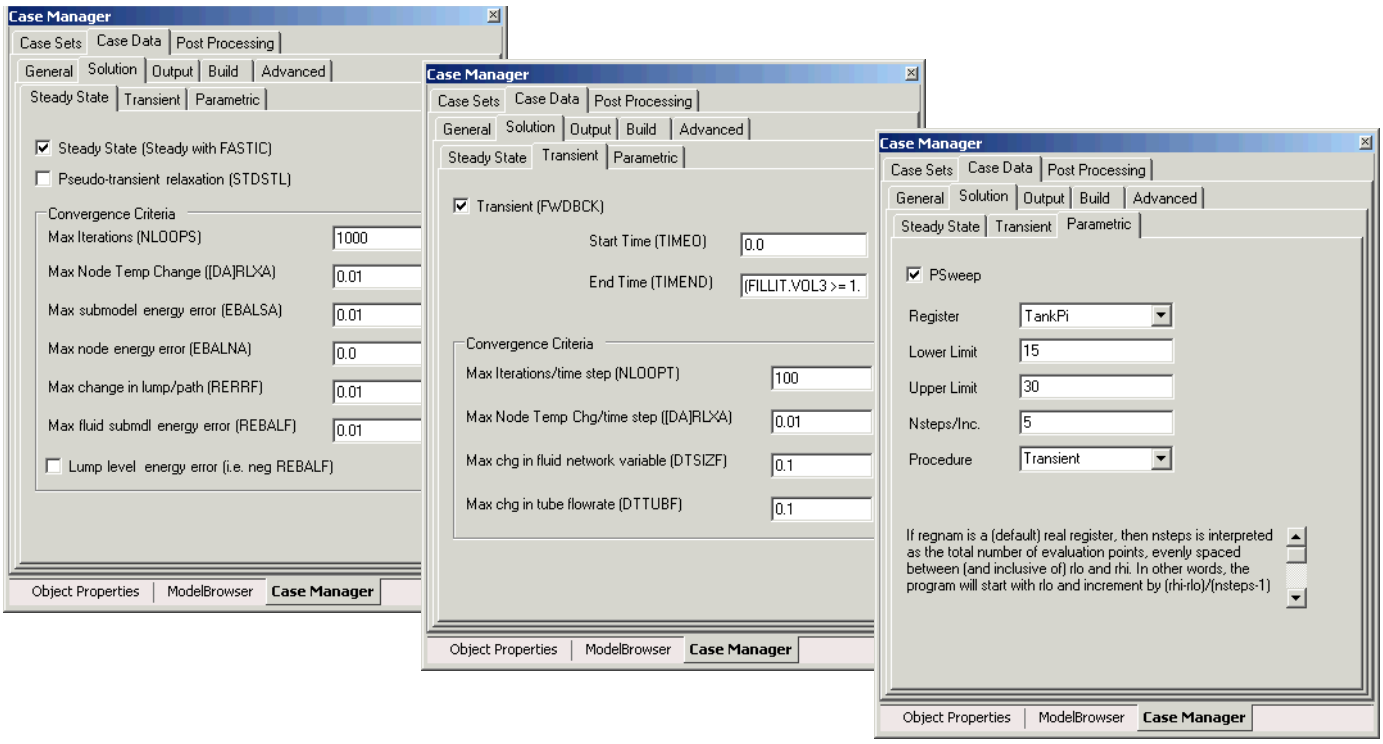
The Solution tab allows the user to specify the type of solution (e.g., transient, parametric) to be run in the case. The solution may be a steady state, transient, parametric, or some combination of these, or possibly a Solver (optimization or test data correlation) run. For steady state, transient, or parametric runs, tabbed forms are available to provide access for specifying the most common control parameters as shown in Figure 9-3. When data is input on any of these solution tabs, Sinaps will automatically create the required logic in the Operations block (“Global Logic Blocks”, Section 5.6.5) in addition to updating Global Control Data inputs.

Multiple solutions can be selected. For example a steady state can be launched followed immediately by a transient or a parametric run. This type of sequencing of runs will enable the results from one solution to be used as the initial conditions for the second solution. If a transient is run after a steady state, the steady state results will be used as the initial conditions for the transient. By default the solution sequence will be written out as steady state followed by transient, followed by parametric (solution tab order left to right). Sequencing multiple runs using the same initial conditions, will require saving of initial conditions and restoring these conditions at the start of each solution.

Alternatively a custom sequence can be created, perhaps steady, followed by parametric, followed by transient, or possibly a transient with initialization logic. In these situations a custom operations data block will need to be created using the Advanced tab of the Case Manager (see “Operations”, Section 9.3.5.80.) To flag to the user that a custom solution sequence has been defined, the solution tab will gray out when a custom operations block has been created.

To invoke the advance Solver or Reliability Engineering modules, the user will need to create a custom Operations block (see “Case Data - Advanced Options”, Section 9.3.5).

FIGURE 9-3. Setting the Case Solution



Steady State Solutions: When defining a steady state case, two options are available to the user: the standard Steady State or a Pseudo-transient Relaxation. In the Steady State solution routine (FASTIC, or "STEADY"), a final solution is found in which all energy and mass flow balances resulting in a time-independent state with no storage of energy, no storage of mass and no fluid momentum; i.e. a true steady state condition. All network objects are assumed to respond instantaneously thus diffusion nodes will act like arithmetic nodes, tanks will act like junctions, and tubes will act like Stubes. Ifaces, twinned tanks, and similar advanced options do not exist in this time-independent solutions.

For most thermal and fluid models, the Steady State option will find a quick solution. However, some fluid submodels need to march through fake time in order to approach a steady-state condition, perhaps because of control systems or volume-dependent options. In this case the Pseudo-transient relaxation option (STDSTL) may produce a better solution for many fluid models. Using STDSTL, the diffusion nodes are still assumed to respond instantaneously however fluid tanks and tubes do not, and this can result in less stability than a pure (real) transient with a fake, large end time (TIMEND). The Pseudo-transient Relaxation method may take more iterations to converge than the Steady State method, and oscillations can occur that prevent convergence as well. Be aware that this method can be *very* intolerant to poor initial conditions in the fluid submodel so it should be preceded by a steady state run (checking both boxes will run STEADY followed by STDSTL).

It is recommended that Steady State (STEADY) be used for most steady state solutions. To better understand the differences between these routines, it is recommended that the user read Sections 4.2.3 and 4.2.4 in the SINDA/FLUINT User's Manual.

A summary of the convergence criteria is provided below. For more information on convergence control, please refer to the SINDA/FLUINT User's Manual.

NLOOPS - This is the maximum number of iterations to attempt for a steady-state solution. An "iteration" means a solution pass through all the nodes and lumps that are active, though the methods by which that pass is made can differ greatly. Generally, a value greater than 500 is rarely needed for a fluid-dominated problem: if it doesn't converge within a few hundred iterations, chances are it never will given the current settings. The default thermal solution settings, however, can take considerably longer unless matrix inversion methods are elected (set MATMET=12 in Global Control Data) instead of the default successive relaxation methods, in which case only a few tens of iterations are typically needed (for a thermally-dominated problem). MATMET=12 (single matrix) is the default for new models, but older models may be set to the prior default of MATMET=0 (purely iteration thermal methods).

DRLXCA/ARLXCA - This is one criterion for accepting a converged solution: node temperatures that do not change by more than this amount per iteration. The default is 0.01°F or 0.01°C... its magnitude is unit-dependent. This default is appropriate for most steady-state solutions, but should be lowered (perhaps to 0.001) for transients and Solver runs (to minimize computational noise).

EBALSA and EBALNA - This is another set of criteria for accepting a converged solution that is generally more important that DRLXCA/ARLXCA. EBALSA is the fractional imbalance allowed in system energy flows. The default of 0.01 means that the energy flowing into the thermal networks should be within 1% of that flowing out of them. EBALNA is a unit-dependent alternative (or complement) to EBALSA that is off by default. It represents the maximum power imbalance (e.g., Watts or BTU/hr) allowed at any one node. EBLANA is rarely used.

RERRF - This is a fluidic criterion for accepting a converged solution that is analogous to DRLXCA and ARLXCA, except that it is expressed as a unitless fraction. The default of 0.01 means that no key fluid parameter (pressure, flow rate, etc.) should change by more than 1% to be considered converged. Sometimes a fluid submodel will be able to converge at slightly larger values (say, 0.02 to 0.05) but not at small values. However, a small value should be used when running steady states under the Solver to help contain computational noise.

REBALF - This is the second fluidic criterion that is analogous to EBALSA, except that the imbalance is checked fluid submodel by fluid submodel. The default of 0.01 means that the energy flowing into any one fluid networks should be within 1% of that flowing out of it. A lump-by-lump check can be applied instead by checking the box at the bottom of the form (lower left of Figure 9-3), which is useful to prevent "STABLE BUT UNBALANCED" messages that are commonly caused by using nondefault path duplication factors (DUPI, DUPJ).

Transient Solutions: To run a transient solution, simply check the Transient (FWDBCK) box and provide a start time and end time for the problem. The transient tab provides access to the most common control constants such as maximum iterations per time step, maximum allowable node temperature change per time step, maximum change in any fluid parameter, and the maximum allowable flow rate change in a tube. Additional control constants are available through Global Control Data ("Case Data - Advanced Options" on page 9-13).

The default transient solution routine* (FWDBCK, or "TRANSIENT") performs a transient thermal analysis by implicit second-order ("forward-backward") differencing in time, and performs fluid analysis by co-solved implicit first-order (backward) differencing in time (see Section 4.2.2 of the SINDA/FLUINT User's Manual for detailed information on this solution method).

* Although not built-in to Sinaps, an alternate transient solution (explicit differencing in time) is available to the user (see Section 4.2 of the SINDA/FLUINT User's Manual), but is rarely superior to TRANSIENT.

If a both a steady-state solution and a transient solution are chosen, the steady-state solution will be performed first, and so will serve as the initial condition for the subsequent transient.

Time-step controls are available, as described below. For more information on time step control, please refer to the SINDA/FLUINT User's Manual. Note that time step control is performed automatically by SINDA/FLUINT, although the user can adjust its calculations as needed.

NLOOPT - This is the maximum number of iterations to attempt for a convergence of a single time step. An "iteration" means a solution pass through all the nodes and lumps that are active, though the methods by which that pass is made can differ greatly. Generally, a value greater than 100 is rarely needed, and is in fact a signal that an excessive time step is being attempted. Nonconvergence at a time step will cause the automatic time step predictor to decrease the attempted step size, so adjustment of NLOOPT is rarely required.

DRLXCA/ARLXCA - This is one criterion for accepting a converged solution: node temperatures that do not change by more than this amount per iteration for time step closure. The default is 0.01°F or 0.01°C ... its magnitude is unit-dependent. This default should be lowered (perhaps to 0.001) for transients,* since each step will otherwise accrue perhaps excessive error.

DTSIZF and DTTUBF - These parameters influence the time step required by the fluid solution. The default of 0.1 for DTSIZF means that no key fluid parameter (e.g., pressure, quality) should change by more than 10% per time step, with tighter controls applied automatically at certain transitions and discontinuous events (e.g., the onset of boiling) yet still in proportion to DTSIZF. Generally, a value beyond 0.5 should not be attempted: a network change (change of assumption) is warranted instead. Similarly, DTTUBF is a specific control for tube flow rate. It is separate from DTSIZF because sometimes tubes are used instead of Stubes for stability but not accuracy,† in which case a large value (typically DTTUBF=0.5, or 50% flow rate variation per time step) can be set to help avoid resolving smaller changes in tube flow rates than would otherwise be required.

Parametric Solutions: The Parametric tab provides a built-in option for creating a parametric sweep:‡ varying any single register over a specified range. These parametric sweeps are useful for running sensitivity studies to assess a system's response to variations in any input. To invoke a parametric run, simply check the box labeled Parametric Sweep and then use the pull-down field to specify which register is to be varied. Minimum and maximum values must be specified along with the number of increments (NSTEPS). Lastly, the type of solution to run must be specified for each value of the register: steady state, transient, Solver procedure (PROC), or reliability procedure (RELPROC).

If one of the other Solution tabs (steady or transient) is also checked, it will be executed prior to running the parametric. For example, in order to run a transient parametric sweep that needs to start from a steady state condition, be sure to select one of the steady state options on the Steady State tab *and* the parametric option on the Parametric tab, selecting the transient procedure option for the parametric.

---

* This default represents a historical value, and this control is also used for steady-state solutions, for which the default is more appropriate. Therefore, the software does not automatically adjust this parameter downward for transients, and instead requires the user to consider doing so.

† As an example, two-phase tanks should not be used in combination with adjacent Stubes. Tubes should be used in that special case to prevent artificial oscillations due to a collision of assumptions.

‡ The parametric sweep is performed using the PSWEEP subroutine in SINDA/FLUINT. Similar design variable and random variable sweep routines are available. Please see DVSWEEP and RVSWEEP in the SINDA/FLUINT User's Manual.

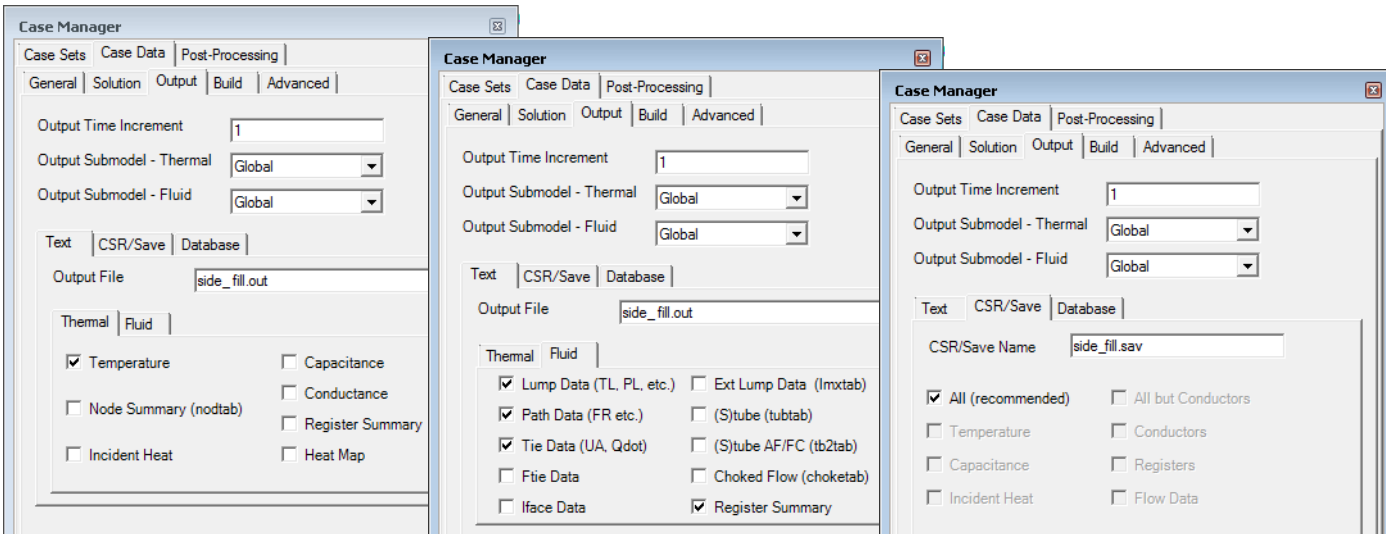
If a real register is chosen (which is the default type), then Nsteps is interpreted as the total number of evaluation points, evenly spaced between (and inclusive of) the lower and upper limits (*rlo* and *rhi*, respectively). In other words, the program will start with *rlo* and increment by (*rhi-rlo*)/(Nsteps-1) until it reaches *rlo*. Nsteps should be more than one in this case. If instead an integer register is named, then Nsteps is interpreted as the increment: a value of 1 causes each point to be evaluated (*rlo*, *rlo*+1, *rlo*+2, ... *rhi*), and a value of 2 causes every other point to be evaluated (*rlo*, *rlo*+2, *rlo*+4, ... *rhi*). Nsteps should be positive for the normal case where rhi>rlo. In the rare event that a reversed direction is needed (i.e., start with a high value and end with a low value), then specify both *rlo>rhi* and negative Nsteps.

Unlike steady state analysis, transients require specific initial conditions, and hence repeated transient analyses require that the network be reset to the same initial condition. If 'transient' is selected as the procedure name for the sweep, then the code internally resets the network (and TIMEO/TIMEN values) to the conditions as they were at the start of the parametric run. This is performed using SAVPAR and RESPAR (saves and restores *all* parameters, except the swept register and any expressions based on that register). If this action is not appropriate, the parametric option should not be used and the user should write their own parametric variation logic in Operations ("Case Data - Advanced Options" on page 9-13), perhaps using a DO loop ("Fortran and Compiling" on page 5-129).

9.3.3 Case Data - Output

The Output tab is where the user can specify what type of output should be generated during a run and how it is to be written. By default, two files are created during a run, a text output file [casename].out and a binary format save file or folder [casename].sav or [casename]_csr. For in-depth information on controlling the processor output, please see "Output and Post-processing" on page 10-1.

FIGURE 9-4. Case Manager Output Options



9.3.4 Case Data - Build

A master model may contain multiple submodels, both thermal and fluid. When a case is defined through the Case Manager, a build configuration must be specified. The ***Build Configuration*** is the set of submodels (often consisting of all submodels) which will be active during the solution. By default, all submodels in the master model are "built" (i.e. included in the solution). If the user only wishes to build a subset, the Build tab (shown in Figure 9-5) provides the abil-

ity to customize the build statements. The pull down build option menu (highlighted in red) along of each section of the form provide multiple options to the user: Build All (the default), Select Below, User Written, Build None or No Build Written.

The Select Below option allows the user to specify which submodels are to be built directly on the form. Simply click on any submodel name to highlight and then click the left/right arrow buttons to send it to the Don't Build Submodels list or back to the Build Submodels list.

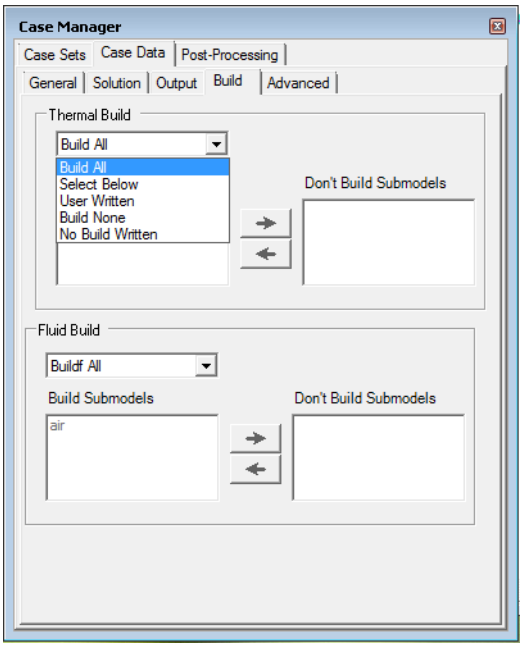
> Note: Once the build configuration is defined for a case, it can be changed dynamically during the solution by the user through additional BUILD or BUILDF statements in Opera-

The User Written options primarily supports import of text based input files where the build statement is already defined in an operations block. When toggled to user written, *both* the thermal and fluid builds will change. When this option is selected, the BUILD (thermal submodels) and/or BUILDF (fluid submodels) statements must be provided in Operations (see Customizing Operations in Section 9.3.5, with an example following in Dynamically Changing the Build Configuration). BUILD and BUILDF statements must start in column 1, including the shortcuts "BUILD ALL" and "BUILDF ALL." Note that if temperature dependent thermophysical properties have been defined in a model and the User Written build option is selected, the user is also responsible for including the "TPDATA" submodel in the build statement. This submodel contains the thermophysical property data. Built configuration names are limited to 32 characters.

The Build None option uses SINDA/FLUINT's Build none option. In this case, Sinaps writes the build statement out as BUILD NONE or BUILDF NONE.

The No Build Written option tells Sinaps to not write the build statement. Note that if the operations block contains a user defined BUILD statement, it will still be written to the input file when No Build Written is selected. This option differs from User Written in implementation. "User Written" toggles *both* the thermal and fluid options to user written, whereas No Build Written toggles only the build statement (thermal or fluid) it is selected for, thus allowing the other build statement option to remain selectable.

FIGURE 9-5. Submodel Build Customization



Dynamically Changing the Build Configuration. Regardless of how the build statement is defined on the Build tab, builds can be changed dynamically during a case by customizing the Operations block. The Build tab settings define the *initial* build configurations for a case. This configuration can be redefined at a subsequent point in Operations logic. Note that the default configuration names in Sinaps are *thermal* for the thermal submodels and *fluint* for fluid submodels. If a build configuration is to be redefined, the same configuration name must be used.

Build statements are erased and replaced (not appended) as new build statements are defined. For example, the Operations logic shown below builds three thermal submodels as the configuration identified as *myconfig*. A steady state simulation is then run, followed by a transient. At 6500 seconds, the simulation stops and control is sent back to Operations. The portion of the model represented by the *stack* submodel is removed from the build configuration by defining a new BUILD statement and a fluid submodel is brought online through a fluid build statement (BUILDF). (In such a case where a fluid model is brought online, additional logic will be required to initialize temperatures and pressures before resuming the transient solution.) While the *stack* model is not built, the temperatures will be held in their last state, and any logic or output or controls specific to the *stack* model will be ignored. If the model is rebuilt through another new build statement (not shown in the example below), the transient response for submodel *stack* would continue from its last active state (in this case, the state it was in at 6500 seconds) unless additional user logic is created to re-initialize the *stack* temperatures to updated values.

```
HEADER OPERATIONS
BUILD MYCONFIG, THERM, WALL, STACK
              CALL STEADY
C
              TIMEO= 0.0
              TIMEND = 6500.
              CALL TRANSIENT
C
C  unbuild stack (separated from vehicle) and build cooling loop fluid model
BUILD MYNEWCON, THERM, WALL
BUILDF LOOP
```

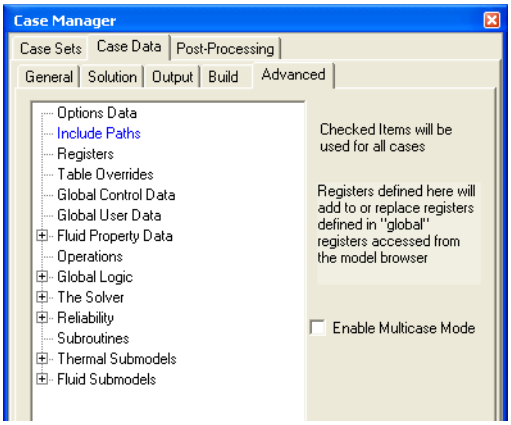
```
C
            TIMEND = 15000.
    (....additional logic to initialize the cooling loop submodel ....)
            CALL TRANSIENT
```

Note: In general, fluid submodels should not be built with zero flow rate. When a complex thermal/fluid system is modeled in which the fluid system turns on/off depending on thermal conditions, it is best to use the dynamic build methods. Build on the thermal portion of the model initially. When the transient conditions are correct for the fluid system to come online, stop the solution, define the new build configurations, and then resume the

9.3.5 Case Data - Advanced Options

The Advanced tab provides access to the standard logic, mixture, and user subroutine blocks. Additionally it provides auxiliary access to other data such as control data, case-specific registers, table overrides, options data, etc. Much of the information traditionally defined in these blocks is automatically defined through other data input within the Case Manager and does not need to be defined here. For example, users can define common control constants on the Solution tab of the Case Manager, or they can go directly to the Advanced tab and access *all* (both common and advanced) control parameters.

FIGURE 9-6. Case Manager Advanced Tab



To access any block from the case manager, simply click on the block name and the block will open in a separate window (most likely tabbed with the network window). For all blocks except Global Control Data, the entry of data will toggle the name of the block in the Advanced tab view to blue, a visual indication of custom content.

The layout of the advanced form is depicted in a tree format, similar to the network tree in the Model Browser. However while the model browser provides access to global registers, tables (arrays), and network diagram information, the Case Manager tree contains everything else in the master model, such as logic blocks, control data, etc. Where applica-

ble, the tree is organized based on global and submodel level inputs as depicted in the table below. Global blocks apply to the master model (all submodels) while submodel level blocks only apply to the submodel in which they reside.

Global Inputs	Thermal Submodel Inputs	Fluid Submodel Inputs
Options Data	Network Includes	Network Includes
Include Paths	Control Data	Control Data
Registers - case specific	User Data	User Data
Control Data	Carray Data	Carray Data
User Data	Source Data	Source Data
Fluid Property Data	Variables 0	Flogic 0
Operations	Variables 1	Flogic 1
Global Logic	Variables 2	Flogic 2
Solver	Output	Output
Reliability		
Subroutines		

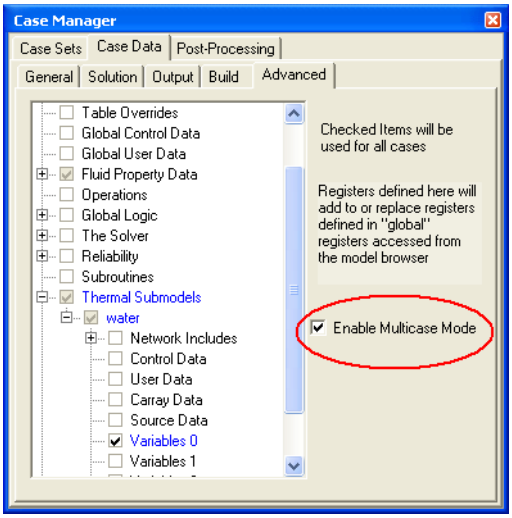
The tree can be expanded or collapsed using the [+] icons to the left of each category. To access any particular entry, click directly on the name and the appropriate dialog will open. In many cases, this will be a simple text editor window. The Advanced form is designed such that any option in the tree containing custom inputs will appear blue (as shown for Include Paths in Figure 9-6).

Each of the advanced options is discussed in further detail below. In general, only a brief description is given since, by the nature of the title "Advanced," these are options that often will require the more complete documentation offered in the SINDA/FLUINT User's Manual.

Application to Multiple Cases. Sinaps provides a "multi-case" option which can be enabled by checking the "Enable Multicase Mode" check box. Without this feature, when updating logic *common to all cases*, the user must edit the block in every case. Insuring such changes are fully propagated through all cases, can be tedious and error prone.

> Note: As an alternative to using the all-case option for logic in the case manager, any logic defined as network element logic will automatically be applied to all cases since the network applies to all cases. For more information on network element logic, see "Network Element Logic", Section 5.6.8.

FIGURE 9-7. Case Manager Advanced Tab



When Multicase Mode is enabled, “all-case” check boxes will appear to the left of each entry in the advanced tab (shown in the graphic below). If the Enable Multicase Mode box is checked and the user unchecks the box, the all-case usage boxes will be reset to the default and will be hidden. By default, the Fluid Includes block is the *only* block applied to all cases (indicating all cases will use the same fluid include files).

When one of all-case boxes is checked, the contents of the corresponding block will apply to all cases in the master model. In the example show above, the Variables 0 all-case box is checked indicating the content will apply to all cases. If edited from *any* single case, the Variables 0 logic will be applied to *all* cases. This is useful for managing subroutine data, and logic blocks common to all cases. Note that checking the all-case box will result in overwriting content which may have been previously defined in that particular block within other cases.

When the all-cases box is toggled from checked to unchecked, the contents of the block in the current case being edited will be propagated to all other cases. Any subsequent changes to a block will apply only to the case where the change was made.

9.3.5.1 Options Data

The Options Data block (Figure 9-8) is where the user defines a model title (essentially a comment of up to 72 alphanumeric characters, which is written at the top of each page in the output file), an optional model name, and output and input files names for program usage. The Options Data form also enables a variety of tools such as mixed arrays, spell checker, etc. The Options Data edit form is designed such that when the cursor is held over any data field or check box, a description of the required input is provided in the lower portion of the window. When updated in the Options Data form, the associated data fields on the General Tab for output file and save file are automatically synchronized.

The only mandatory input is the title, which is defaulted to the case name. The user can override this default and change the title (which does not affect the case name). All other data fields are optional and defaulted if appropriate.

The Restart Output (RSO) file is a file written in binary format during the solution, similar to a SAVE file, allowing a user to store data for use in a different case. The routine RESAVE must be called from user logic to write the RSO file.

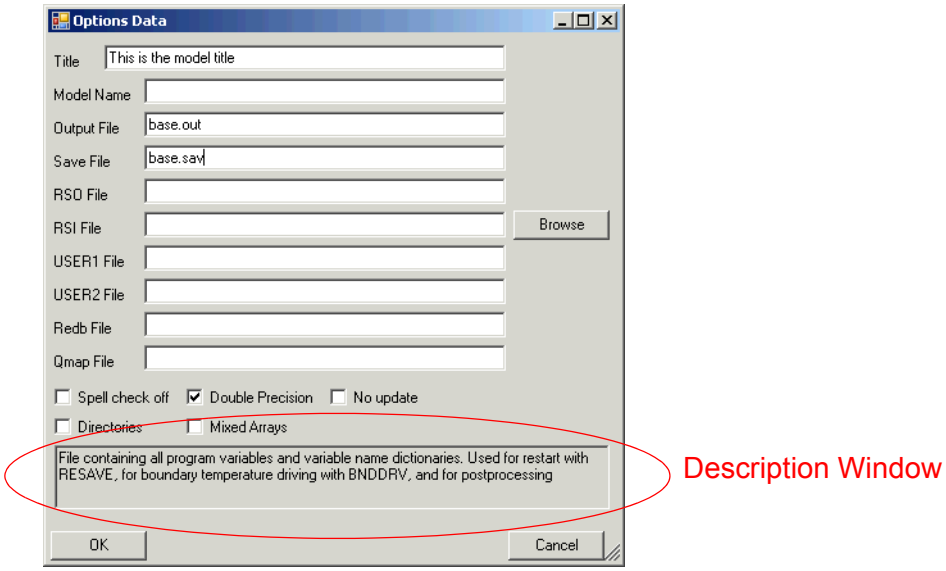
The Restart Input (RSI) file is a *previously* written RSO file (i.e., from an earlier run), possibly from a different case, which contains data to be read into the model for setting up initial conditions, setting registers, etc. An RSI file is read into a model using the RESTAR subroutine.

The user may optionally define two user files: USER1 and USER2. These files can then be written to from any logic block by referring to those files as unit numbers NUSER1 and NUSER2, respectively in a write statement, such as:

```
WRITE(NUSER1,*)  ' TEMPERATURE OF NODE 15 = ', MYSUB.T15
```

Note: If more user-defined files are required, they may be opened by calling the USRFIL or USRFIL2 routines (see Section 7.4.16 of the SINDA/FLUINT User's Manual).'

FIGURE 9-8. Options Data



The REDB File supports the Reliability Engineering module. This is yet another binary results file that uses exactly the same format as the SAVE and RSO files, used for either restarting or post-processing. Uniquely, the REDB file can also be used to perform saves and retrievals within the same run. More importantly, it can be used to add new data to the results of an old run. For more information on the REDB file, please see section 5.23of the SINDA/FLUINT User's Manual.

The QMAP file supports writing the results of the thermal network Heat Map (see "Case Data - Output", Section 9.3.3) to a file other than the main output file. If a QMAP file name is defined in Options Data, the map will be written to this file in addition to the output file.

The check boxes at the bottom of the Options Data form provide the ability to toggle various features summarized in Table 1.

TABLE 1. Option Data Toggle Descriptions

Option	Description
Spell Check Off	Turns off user logic spell checker (not recommended)
Directories	Print flag for node, conductor, array, lump, path, tie, and user constant directories (rarely used)
Double Precision	Directs that key program variables are stored and calculated used using double precision (recommended: leave checked)
Mixed Arrays	Turns off checks of mixed real and integer types within arrays
No Update	Turns off the automatic updating of expressions using registers and processor variables (not recommended)

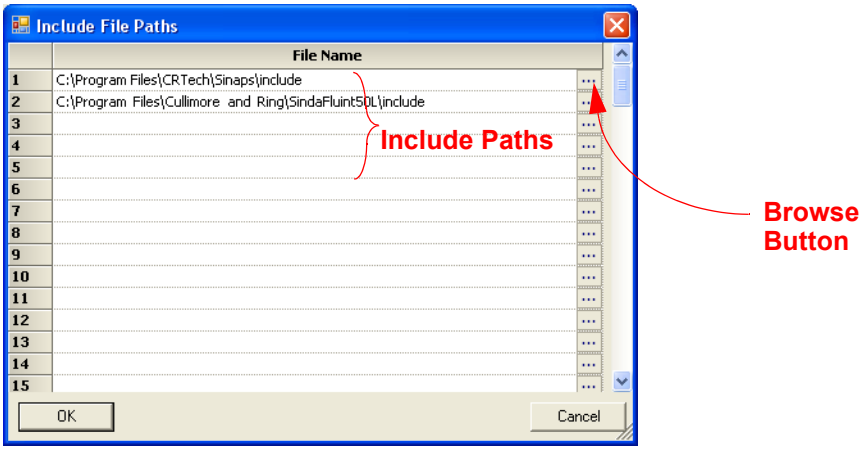
9.3.5.2 Include Paths

The Include Paths option allows the user to define where the code should look for INCLUDE files. By default, it will look in the working directory first. Additionally, the default input in the Include Paths form specifies the program to look in the Sinaps installation directory. The user can define additional paths the program should search for INCLUDE files by clicking the browse button and locating the directory to search. This will result in the path being added to the Include File Paths form. If multiple files with the same name exist in different directories (which have all be tagged to be searched), the program will use the first file it finds based on the defined sequence of paths in this form.

> **Caution:** Use of this form will overwrite any *paths.txt* file that the user may have placed in the model's current subdirectory.

Note that paths.txt can also be placed in the SINDA/FLLUINT install directory. This can be useful if a "library" of fluid descriptions is keep somewhere other than the model working directory. For more information on the paths.txt feature please see section 2.5.2 of the SINDA/FLUINT User's Manual.

FIGURE 9-9. Include Paths



9.3.5.3 Registers

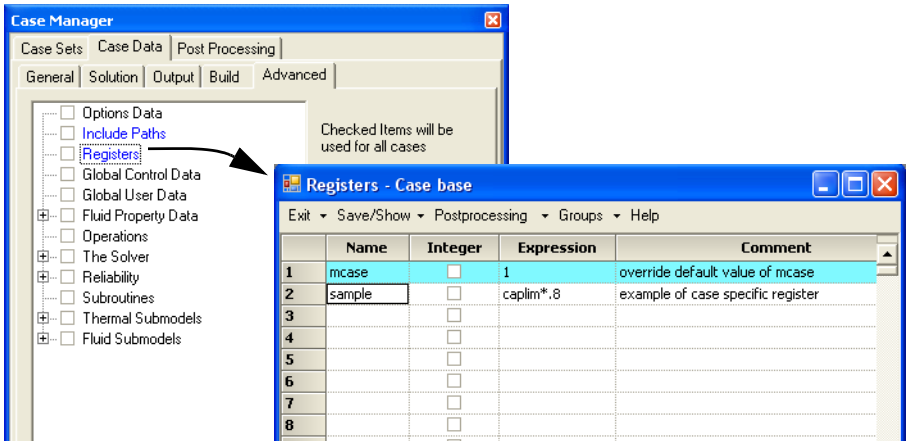
The Registers option provides the ability to define case-specific registers. These can base used to define a case level overrides for global registers or for defining new case-specific registers for use in logic and control. *Using case-specific registers in network object definitions, it not recommended*. If a new case is subsequently added without defining the case-unique register, the resulting model will fail in the preprocessor. On the other hand, global register overrides can be used to define network objects. The case-specific register form is shown in Figure 9-10. These registers will either replace or (if uniquely named) be added to the set of Global Registers when the case is run.

> **Caution:** When using restart operations between two cases, case level register overrides can be used however both cases must contain the same set of registers. No case specific registers can be used unless they are defined in both cases.

To define a unique case based register, simply provide a name, defining value or expression, and optional description. Like global registers, a case specific register can be defined as an integer by selecting the Integer check box. When creating a register override, the list of global register names can be accessed by placing the cursor in the name cell and clicking the middle mouse button to bring up a Global Register menu option. Clicking this option will provide a selectable list of existing global registers. When a global register override is defined, the input row in the case specific register form will highlight in cyan identifying the input as an override versus a case-specific register.

See “Parameterizing a Model” on page 1. for more information on using case-specific registers, specifically “Case-specific Registers” on page 6-7.

FIGURE 9-10. Case Based Registers and Global Register Overrides



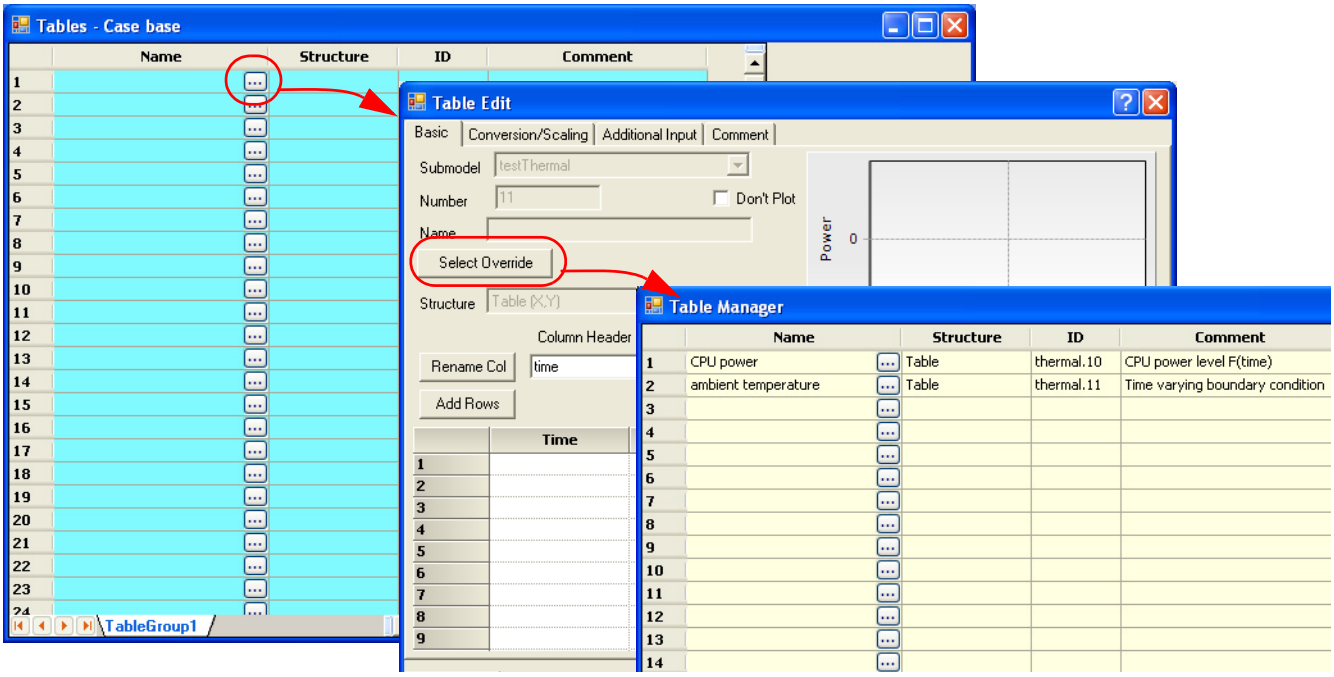
9.3.5.4 Tables

Any Sinaps table can be defined to be case specific, i.e. the data contained can vary from case to case. Note that when defining a case level override of a table, the name, submodel, and ID of the table must remain the same, only the data changes. The table structure and the number of entries in the table *must* remain consistent with the original table.

To create a case level override of a table, select Tables in the advanced tab dialog of the case manager to bring up the Table Override Manager, shown in Figure 9-11. The override manager background is blue to visually distinguish it from the main Table Manager dialog. Only overrides can be created in this dialog, new tables cannot be added. Click on

the [...] button to create a table override. The Table Edit dialog will appear. Click on the Select Override button to bring up the normal Table Manager and browse the manager for the table for which the override is to be created. To select a table click on the row number and then click OK. The table edit form will update with the correct submodel, number, name and structure. Enter new data in the data table and click OK to save the contents.

FIGURE 9-11. Case Based Table Overrides



9.3.5.5 *Global Control Data*

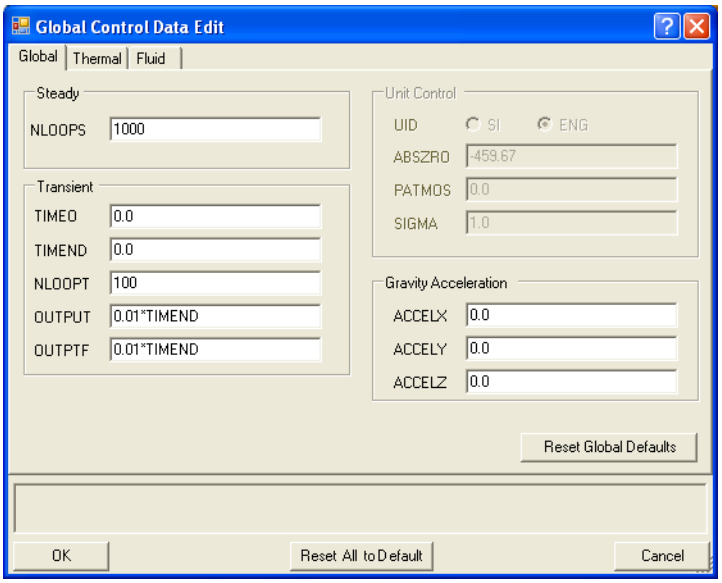
Global Control Data inputs allow the user to control the problem solution by initializing a set of control constants. The term “constants” is a historic misnomer: most (but not all) may be initialized or changed at will in the user’s logic blocks or via defining expressions. The term “Global” is used to indicated the data applies to *all submodels*. Under Thermal and Fluid Submodels on the Case Manager Advanced tab, “Control Data” options are provided to allow submodel level overrides of Global Control Data.

> **Caution:** By default the “all-cases” box for Global Control Data is unchecked, indicating that the contents are case specific. If the user checks this box, the current control data values will be propagated to all cases. Note that when the Global Control Data is applied to all cases, all cases will have the same end time for transients, same output intervals, etc. For more information on the all-cases box, see See “Application to Multiple Cases” on page 14.

Control constants control the solution process (number of iterations, end time, etc.) and define the convergence criteria for steady states and the time step for transient simulations, hence they are typically case specific. Examples of these constants are and, the transient end time TIMEND and maximum steady state iteration count allowed NLOOPS. Where possible, constants are defaulted to reasonable values.

By clicking on the Global Control Data option, the user can access *all* control constants available as shown in Figure 9-12. Tabbed forms are used to divide the control data into global and thermal- and fluid-specific control data (see Figure 9-13 and Figure 9.3.5.6 respectively). The most common of these constants are also accessible elsewhere in the Case Manager, such as on the General or Solution tabs. Note that the Unit Control is grayed out. Unit selection is made when a model is first created. The data is shown here for reference purposes since it is considered control data but is not normally considered case specific and therefore not editable through the case manager. To modify units see “Setting Units for the Model” on page 5-4.

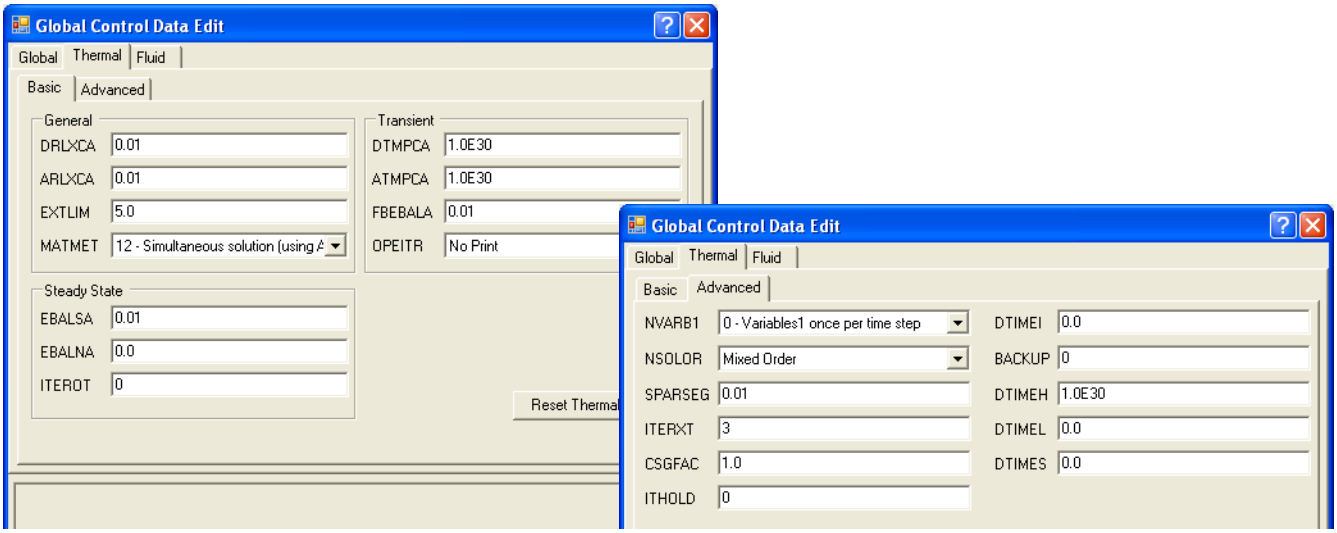
FIGURE 9-12. Global Control Data



By hovering the mouse in any Control Data field, help text will appear in the lower portion of the window providing a more detailed description of the constant. For a full list of all control data variables and details of their usage please refer to Section 4.3 of the SINDA/FLUINT User’s Manual.

The Global tab is further divided into categories such Global, Thermal, and Fluid, along with Basic and Advanced for both the Thermal and Fluid tab.

FIGURE 9-13. **Thermal-specific Control Data (Fluid similar)**



Resetting Control Data. The Global Control Data forms provide buttons to reset constants to the Sinaps defaults. Each tab contains a Reset [Tab Name] Defaults button which will reset the contents of the tab. To reset all constants to their defaults use the Reset All to Default button at the bottom of the form.

9.3.5.6 Global User Data

Global User Data is a historical feature which as been largely replaced by registers. It is provided here for backward compatibility. Users are encouraged to use registers instead of User Data. If an older model containing User Data is read in, the User Data block contents will be placed here.

Clicking on the option will open an editable text window in which SINDA/FLUINT text input rules apply.

9.3.5.7 Fluid Property Data

There are two Case Manager options for fluid property data, fluid property Includes and Mixture Data. The Includes option provides a “model level” list of *all* fluid property INCLUDE files used in all fluid submodels. With this dialog a fluid description may be overridden at the case level. Note that if a fluid is used in more than one fluid submodel, changing it in the case manager will impact all fluid submodels which use that fluid.

When defining a fluid override, the fluid ID (numeric value) in the new include file must be identical to the ID defined via the Model Browser at the individual submodel level. This is useful for testing permutations using different fluid descriptions, for example compressible liquid (CL) or neverliq (NL) property descriptions. For more information on defining working fluids, see “Specifying Fluids”, Section 8.1.4 and “Working Fluids”, Section 5.4.1.

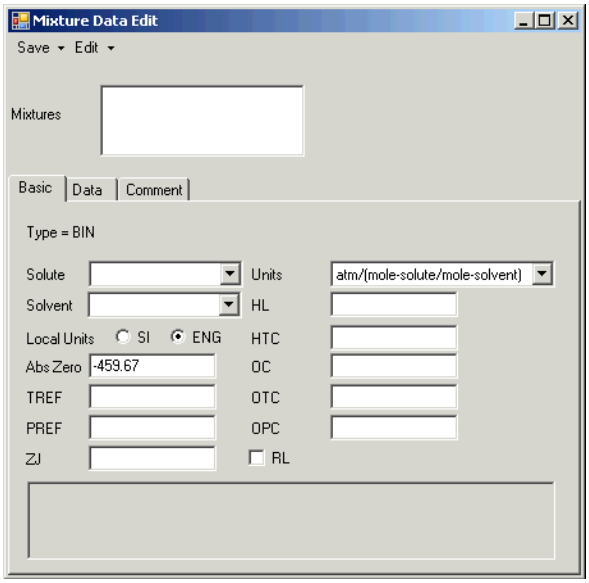
By default the all-cases box is checked, indicating that the fluid include dialog is common to all cases defined in the case manager. If the user unchecks the box, the current fluid includes will be propagated to all cases. Subsequent editing of fluid include data will apply to only the current case.

The term “Mixture Data” is a general term reserved for future expansions; the only currently available Mixture Data option is to define the solubility between two fluid species. Perfect gases and other fluids may be used as species within

working fluid mixtures. By default, a perfect gas species is assumed to have zero solubility. However, if required the program can be used to calculate equilibrium and finite rate dissolution and evolution of noncondensible gases into and out of other liquid species.

To enable dissolution to be modeled, the user must define the equilibrium solubility between any pair of solvents and solutes using the Mixture Data form. Similar to other data input forms, the Mixture Data form provides descriptions in the lower portion of the form when the cursor is held over any data input field. For more information on mixtures see "Working Fluids", Section 5.4.1. For more information in how to define solubility, see Section 3.22 of the SINDA/FLUINT User's Manual.

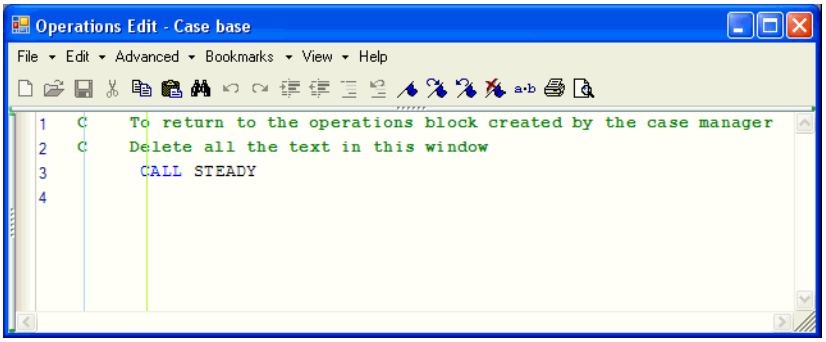
FIGURE 9-14. Defining Mixture Data



9.3.5.8 Operations

The analysis operations to be performed on the network are contained within the Operations block. By default, Sinaps builds this block automatically from user-selected options within the Solution tab of the Case Manager. An example of an automatically generated Operations block is shown in Figure 9-15.

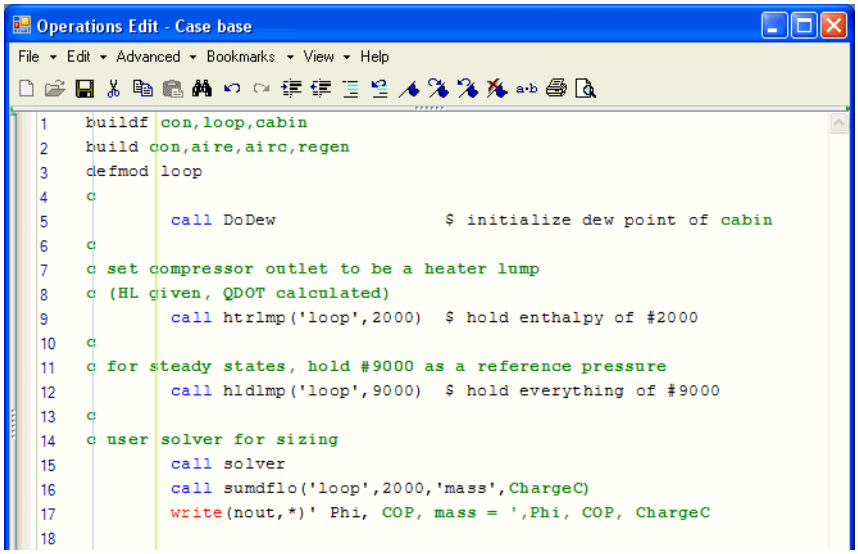
FIGURE 9-15. Example of Automatically-generated Operations Block



The user can override the automatic Operations block by using this advanced option. When the automatic block is overridden, the contents of the Solution tab in the Case Manager will be grayed out indicating a custom Operations block has been created. Once a customized block has been created, the user can return to the automatically-generated block by deleted the contents of the customized Operations block.

Selecting the Operations option on the Advanced tab will open an text window allowing the user to create a custom Operations block, an example is shown in Figure 9-16. In this example, the user has chosen to write their own build statements (User Written option on Build tab of the Case Manager), defining a default model for subsequent logic, adding initialization logic, defining a heater lump and holding a lump, and then calling the Solver as a solution with some follow-up logic to write information to the output file.

FIGURE 9-16. Example of Customized Operations Block


Operations Edit - Case base
File Edit Advanced Bookmarks View Help
1 buildf con,loop,cabin
2 build con,aire,airc,regen
3 defmod loop
4 c
5 call DoDew $ initialize dew point of cabin
6 c
7 c set compressor outlet to be a heater lump
8 c (HL given, QDOT calculated)
9 call htrlmp('loop',2000) $ hold enthalpy of #2000
10 c
11 c for steady states, hold #9000 as a reference pressure
12 call hldlmp('loop',9000) $ hold everything of #9000
13 c
14 c user solver for sizing
15 call solver
16 call sumdflo('loop',2000,'mass',ChargeC)
17 write(nout,*)' Phi, COP, mass = ',Phi, COP, ChargeC
18

The Operations block is executed once during a case: the program terminates when all the operations specified in Operations have been completed. The Operations block contains all logic necessary for setting up initial conditions for the case, specifying the type of solution to be run (steady, transient, Solver, etc.), and any wrap-up logic after the execution is complete. It is also where the user may specify restarts or dynamic configuration builds.

Note that Operations is a “Global” logic block applying to all submodels; therefore all processor variable references must include a submodel model name. The DEFMOD command eliminates the need for the repetitive input of a model name:

```
DEFMOD smn              $ The defmod call should be placed in column one
```

For example, “MODL.T100” could be input as “T100” after a DEFMOD MODL command. The default submodel name may be either a thermal or fluid submodel, but only one submodel name may be defaulted at any time.

For full documentation on the Operations data block, please see Section 4.1.3.1 of the SINDA/FLUINT User’s Manual.

9.3.5.9 Global Logic

In addition to submodel-level OUTPUT CALLS, VARIABLES, and FLOGIC blocks, global level blocks are available that are always called independent of any BUILD or BUILDF status, and which always follow the submodel-level blocks in the processor. For example, any logic placed in the global FLOGIC0 block will be invoked after the FLOGIC 0 blocks for all fluid submodels have been called (including if there are none) every time step or steady-state iteration. Whereas the instructions contained within the global VARIABLES 1 block are executed once per transient time step unless the global control constant NVARB1=1 has been set for all thermal submodels. The instructions contained within global OUTPUT are invoked based on the global output interval defined and *any* submodel's block output interval.

Because there is no default submodel in a global-level block, the submodel prefix for network variables such as T (temperature) and PL (pressure) is not optional, though the DEFMOD command is available as an input convenience.

9.3.5.10 The Solver

The Solver is part of the Advanced Design Modules of SINDA/FLUINT: a high-level utility that iteratively runs steady states and transients as subprocesses toward a larger purpose. That "larger purpose" might be optimizing a design (e.g., sizing), or goal seeking (e.g., solving for what is normally an input given what is normally an output), or calibrating a model's uncertainties given test data, or searching for a worst-case design scenario.

The Solver adjusts one or more ***design variables*** until ***Object*** is as close as it can be to ***Goal***, perhaps subject to various ***constraints***. To estimate how any one design performs, the Solver tests a set of values for the design variables in an ***evaluation procedure***, whose purpose it is to determine how well any one design stacks up against the Object and the constraints. The evaluation procedure, which can be arbitrarily complex, is iterated repeatedly as the Solver tests new design variable values.

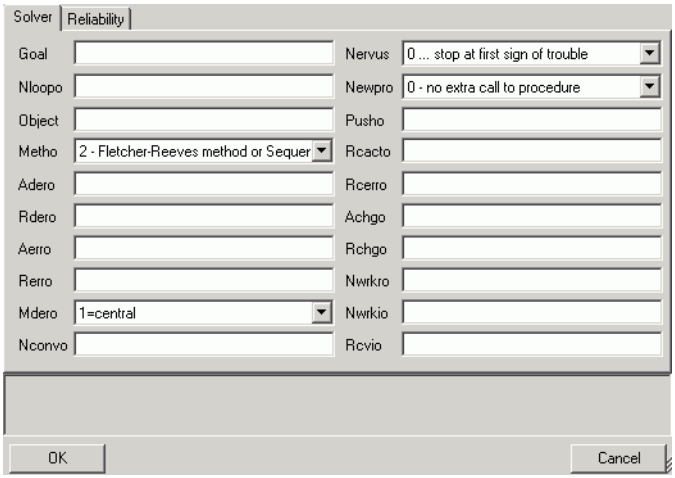
To use the Solver, a custom Operations block *must* be built invoking the Solver as in the following example:

```
build all
buildf all
      call steady               $ get initial conditions
      call svpart('-R', mtest)  $ save to re-initialize a transient in PROC
      call solver                 $ start the Solver working
```

The Solver is fully documented in Section 5 of the SINDA/FLUINT User's Manual.

Solver Data: Figure 9-17 depicts the Solver Data form, with the information window at the bottom providing descriptions of each field as the cursor is hovered over it.

FIGURE 9-17. Solver Data



The ***Goal*** is defaulted to -1.0e30, which is essentially means "minimize Object." To maximize the objective, set Goal to 1.e30. If the goal can be reached, it can be set as a finite number for "goal seeking." For example, if the problem at hand were "How much heat should be added to make the temperature of node 10 in submodel *ristra* reach 230.0 degrees, then the Object would be "ristra.T10" and the Goal would be 230.0 (and the design variable would be the amount of heat to add). Once set, the Goal should not be changed during a Solver run.

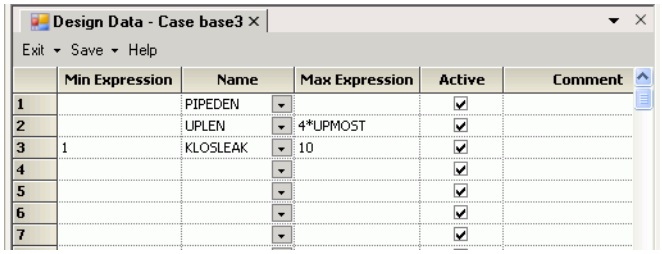
The ***Objective*** (Object) is that which is to be minimized, maximized, or adjusted to meet a finite goal (in the case of goal seeking). If it can be expressed as a simple expression (perhaps including registers which might themselves be design variables), then that expression can be entered in the form. More complex calculations for Object may be made in user logic (usually in Procedure), in which case the Object field can be left blank. For example, if OBJECT (the SINDA/FLUINT variable name) is to represent the root mean square of a comparison between test data and predictions,* OBJECT would need to be zeroed, perhaps hundreds of squares differences summed into it, and then its square root would need to be taken. Such manipulations are beyond what can be accomplished in the expression for Object.

Design Data: ***Design variables*** (or correlation uncertainties, etc.) must first be real registers. One of more of those real registers may then be chosen as Solver design variables using the form shown in Figure 9-18. Design variables are the "degrees of freedom:" those values which the Solver can adjust to make the objective become as close as possible to the Goal, subject to constraints.

The middle column (Name) is a pull-down of available real registers. Once chosen, lower and upper limit to the variation of those parameters may be entered if needed. In the example shown in Figure 9-18, there are three design variables: the first (*pipeden*) has no limits, the second (*uplen*) has an upper limit (an expression, in this case), and the third (*klosleak*) has both upper and lower limits.

* Routines such as COMPARE are available to automate such tasks.

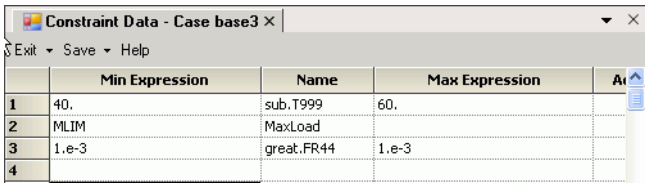
FIGURE 9-18. Design Data



Constraint Data: The Solver can optionally be constrained by additional rules in its search. These rules might distinguish a useful design from an inviable one. For example, the mass of a heat exchanger might be minimized while its performance is no less than some minimum threshold, say 95% effectiveness. The mass is the Object (Goal is defaulted to -1.e30), and 95% becomes a constraint. In the example shown in Figure 9-19, there are three constraints imposed. The first and third "Names" are actually expressions. The first constraint assures that, when the final answer is found, that 40.0 ≤ sub.T999 ≤ 60.0 and the third constraint (an equality constraint) prescribes that great.FR44=1.0e-3 (within tolerances) at the final answer (but perhaps not for all designs tested).

The second constraint assures that MaxLoad is greater than MLIM for the final design. But *MaxLoad* is not a register: in fact, it *can't* be.* Its use spawns the creation of a new type of variable: a named constraint variable. Basically, a named constraint variable is used when the constraint is too complex to be phrased as a simple expression within Constraint Data. Instead, *MaxLoad* must be calculated in user logic (usually in Procedure) in much the same way that OBJECT *might* be.

FIGURE 9-19. Solver Constraint Data



Procedure: The ***Procedure*** is like a custom-built Operations block. In fact, it can contain anything that Operations can contain, except a recursive call to Solver itself. The Solver calls Procedure repeatedly with new values of design variables to test. The job of the Procedure block is to make sure that the Objective and any named or unnamed constraints are updated before the block is finished. Often, Procedure will simply consist of a single steady state call ("CALL STEADY" tabbed out of column 1), but it can be considerable more complicated as needed.

Solver Logic Blocks: Solver logic blocks represent Solver-level opportunities for customization. The SOLOGIC blocks are rarely used, but if needed they are a global (not submodel-specific) logic block accessible by the Sinaps logic editor. A more common block is the SOLOUTPUT block, which might contain calls to output routines such as DESTAB and CSTTAB (which have no arguments), or it might contain calls to SAVE etc. for post-processing the Solver's progress, if needed.

* However, an *expression* such as 1*MaxLoad or MaxLoad+0 is valid in case MaxLoad happens to be a register, and no new constraint variable need be named. Nonetheless, since it is defined by expression and not by a named constraint variable, this constraint is "unnamed."

9.3.5.11 Reliability

Along with the Solver, Reliability Engineering (also known as "statistical design" and "stochastic design") is part of the Advanced Design Modules of SINDA/FLUINT: three high-level utilities that iteratively run steady states and transients as subprocesses toward a larger purpose. That "larger purpose" is answering questions such as "Given uncertainties in certain inputs, what are the chances that a set of operational limits will be exceeded?" Reliability Engineering has many parallels to the Solver, and (despite the long description that follows) easier to apply.

The user starts by identifying which inputs are uncertain (the ***random variables***) and by how much (their ***probability distribution function***). For example, a boundary condition like freestream temperature might be uncertain and its variation might be Gaussian: described by a bell curve whose mean is 34°C with a standard deviation of 6.2°C. Or a thickness tolerance on a plate might make any value between 0.51 and 0.53 mm equally likely (0.52±0.01): a uniform distribution. Or test data might be available for the possible contact conductance variations that are produced by an assembly line. Or a material property such as thermal conductivity might be most likely to be 12.3 W/m-K, but never less that 11.0 W/m-K nor more than 14 W/m-K (a triangular or "witch's hat" distribution).

Next, the user identifies possible failure limits. For example, the user may wish to know what the chances are that the temperature of a particular component will exceed 80°C, or that a valve will shut before 80 seconds, or that a heater will switch on/off more than 100 times. In more general terms, the user may check many responses for many limits, including multiple limits for the same response (e.g., the likelihood that the heater switches more than 80 or 90 times). These response limits are call ***reliability constraints***. (Up front designations are helpful but not mandatory for some statistical methods: the user may apply hindsight by querying different responses after a run has been completed.)

Once the uncertainties and the reliability constraints have been identified, the user must provide an Operations-like procedure which will be sampled repeatedly using different values of the random variables. This reliability procedure is often a simple steady state solution, but can be arbitrarily complex as long as the program knows the values of any important responses (e.g., temperatures, heater switch count, valve shut-off time) by the end of that logic block.

Finally, the user chooses one of three methods to perform the statistical analysis by choosing a routine to call from Operations (summarized in Table 2):

SAMPLE - This is a Monte Carlo sampler: a "shotgun approach." It performs up to NLOOPR calls to the Reliability Procedure, using randomized values of the random variables (subject to their probability of occurrence). It monitors the responses defined as reliability constraints to determine convergence. SAMPLE is expensive: it typically takes 1000 to 3000 calls, which makes it intractable for all but the simplest problems. However, it is also the most powerful method in terms of applicability, appendability (for multiple runs), etc.

DSAMPLE - This is a descriptive sampler. It performs exactly NLOOPR calls to the Reliability Procedure, using quasi-randomized values of the random variables. It intelligently subdivides the distribution of each variable into NLOOPR ranges of equal probability, sampling at the statistical center of each such division. The exact values that will be sampled at any one time are randomized, so that repeated runs will not yield exactly the same result. DSAMPLE is approximately 5 to 10 times more efficient than is SAMPLE, meaning it can achieve the same numerical accuracy in 1/5^{th} to 1/10^{th} of the number of samples. Therefore, set NLOOPR to as high a value as can be afforded, with 100 to 200 samples required to yield adequate accuracy for most problems.

RELEST - This is a quick and highly approximate estimation that is not based on sampling, but rather on finite differencing. Reliability Constraints can be defined after the fact for the sampling methods, but should be defined in advance

when using RELEST. RELEST takes exactly N+1 calls to the Reliability Procedure, where N is the number of random variables. (Therefore, make sure NLOOPR is set to N+1 or greater.) RELEST makes a lot of assumptions to do its job so quickly, including assuming that all responses are linear when deviated off the initial point, and that both the random variables and the responses can be approximated locally as Gaussian. (Random variables need not be normal distributions to use RELEST.) Despite these approximations, RELEST can get close to the true answer with a very small expense. It is intended either for rough estimates, or to be used as a subprocedure of a reliability-based optimization. For example, using RELEST nested under the Solver's Procedure block, the user can size both a design variable *and its allowable tolerance* at the same time. (See Section 5.27, Reliability-based Optimization, in the SINDA/FLUINT User's Manual for details and examples.)

TABLE 2. Comparison of Reliability Estimation Routines

	SAMPLE	DSAMPLE	RELEST
Method	Monte Carlo sampling	Descriptive sampling	Gradient method
Speed	Slow	Intermediate	Fast
Convergence Detected?	Yes	No	No
Fixed Execution Cost?	No	Yes	Yes
Overall Reliability?	Yes	Yes	No
Cumulative?	Yes	Somewhat	No
Applicability?	Unlimited	Unlimited	Limited. Assumes: - NORMAL variables - Linear responses - Continuous responses - Fixed failure limits
Incomplete Runs Tolerated?	Yes	No	No
Notes		Can't change NLOOPR while executing.	NLOOPR .GE. N+1, where N is the number of random variables.

The routine RANTAB provides an echo of inputs for the random variables that are defined, while the RCSTTAB provides outputs for the samplings made: an estimate of the chances that the reliability constraints (or "failure limits") will be exceeded.

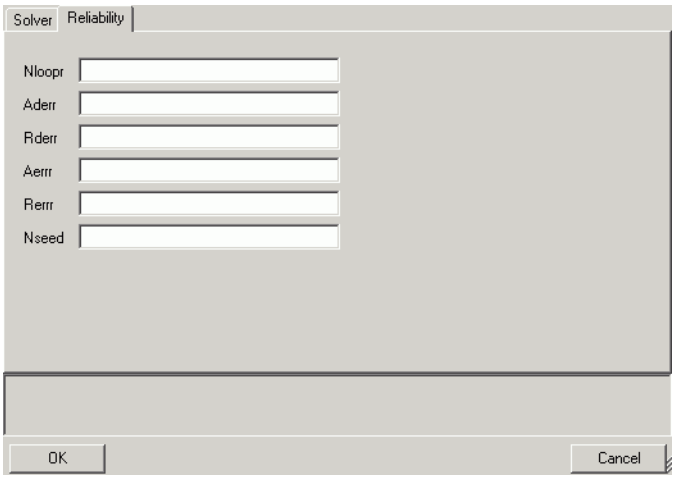
To use statistical sampling, a custom Operations block *must* be built invoking one or more of the sampling routines (RELEST, DSAMPLE, SAMPLE) as in the following example:

```
build all
buildf all
      call rantab          $ echo Random Variable info for checking
      call dsample         $ perform NLOOPR descriptive samples of RELPROCEDURE
      call rcsttab           $ tabulate
```

The Reliability Engineering Module is fully documented in Section 5 of the SINDA/FLUINT User's Manual.

Solver Data: Figure 9-20 depicts the Solver Data form with the Reliability tab selected. As usual, the information window at the bottom providing descriptions of each field when the cursor hovers over it. The key input is NLOOPR, which is either the number of samples to take (for DSAMPLE) or the maximum number allowed (for SAMPLE).

FIGURE 9-20. Solver Data (Reliability Data)

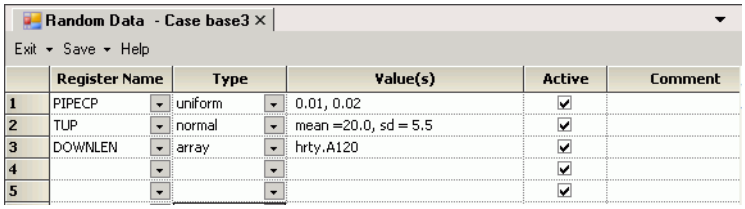


Random Data: ***Random variables*** are uncertainties whose variation is to be sampled by the appropriate reliability estimation routine (RELEST, DSAMPLE, or SAMPLE). Random variables must first be real registers. One of more of those real registers may then be chosen as random variables using the form shown in Figure 9-21. Three types of variations may be defined: ***uniform*** (any value equally possible within a range), ***normal*** (a Gaussian or bell-shaped curve of probability defined by a mean and a standard deviation), and ***array***. The last method permits any other type of ***probability distribution function*** to be defined using a doublet SINDA array, the specific values of which are not important since they will be normalized. The array method might be used to represent a truncated bell shape, test data, or a simple "witch's hat" (triangular) distribution that, while simplistic, is still more realistic than a uniform curve.

The first column (Register Name) is a pull-down of available real registers. Once chosen, the user must define the type of variation (uniform, normal, or array). Depending on the type, text inputs complete the definition. For the uniform distribution, comma-separated lower and upper limits (perhaps expressions) can be entered. For the normal distribution, key-word driven inputs define the mean (*mean*) and the standard deviation (*sd*, though the coefficient of variance, *cv*, may alternatively be supplied.) For the array input, a reference (complete with submodel prefix) to a Sinaps XY table (SINDA doublet array) should be provided. The array should contain V_1, P_1, V_2, P_2 ... where P_i is the relative probability of value V_i for that register.

In the example shown in Figure 9-21, there are three random variables. The first (*PipeCp*) can vary from 0.01 to 0.02 with equal (uniform) probability of any value in between. The second (*Tup*) has a normal (Gaussian) variation with a mean of 20.0 and a standard deviation of 5.5. The third (*DownLen*) is an array variation as supplied by the SINDA array #20 in submodel *hrty*.

FIGURE 9-21. Random Data



Relconstraint Data: ***Reliability constraints*** are input in almost an identical manner to Solver constraints (Figure 9-19), except their meaning is very different. Reliability constraints are really failure limits, asking the question: "What are the chances that this variable will exceed this range?" In fact, *relconstraints* do not affect the answer (except maybe the convergence of SAMPLE runs). Rather, they are really queries that can be made *after* a run is completed. For that reason, the user might add many of them, even ones that might at first seem redundant. For example, in addition to checking to see what the chances are that "fraud.T40>10.0," the user might also add many similar limit checks to get more information from the same run such as "fraud.T40>20.0" and "fraud.T40>30.0." Similar post-processing queries can also be made using EZXY®, the CRTech plotting program.

Relprocedure: The ***Reliability procedure***, or *relprocedure*, is like a custom-built Operations block. In fact, it can contain anything that Operations can contain, except a recursive call to the originally invoked sampling routine (e.g., RELEST, SAMPLE, or DSAMPLE) itself. The sampling routines calls RelProcedure repeatedly with new values of random variables to test. The job of the RelProcedure block is to make sure that the relconstraints are updated before the block is finished. Often, RelProcedure will simply consist of a single steady state call ("CALL STEADY" tabbed out of column 1), but it can be considerable more complicated as needed.

Reliability Output Calls: If either DSAMPLE or SAMPLE are used, then the data collected by the sampling routine is the key result, and not necessarily any specific results reported by RCSTTAB (though it always helps to define as many reliability constraints as possible up front). This collected data may be stored in a "reliability database," which is simple another form of SINDA/FLUINT SAVE file which can be post-processed, perhaps creating a histogram or calculating other statistics inside of EZXY, the CRTech plotter.

To store the date, first define an REDB file in Options Data (e.g., "reliability.sav" in the Redb field). Next, in the Reloutput text block, insert a call to SaveDB such as:

```
CALL SAVEDB('ALL',NTEST)
```

The argument occupied above by "NTEST" is an integer record location returned by SAVEDB, as explained in the SINDA/FLUINT User's Manual. It is used for advanced features such as appending to the database.

9.3.5.12 Subroutines

This Advanced tab option provides text edit window for entering user subroutines. User Subroutines are documented in "User-defined Subroutines" on page 5-136. Note that multiple subroutines can be located in this window, and that additional ones may be fetched using embedded INSERT commands.

9.3.5.13 Network Includes

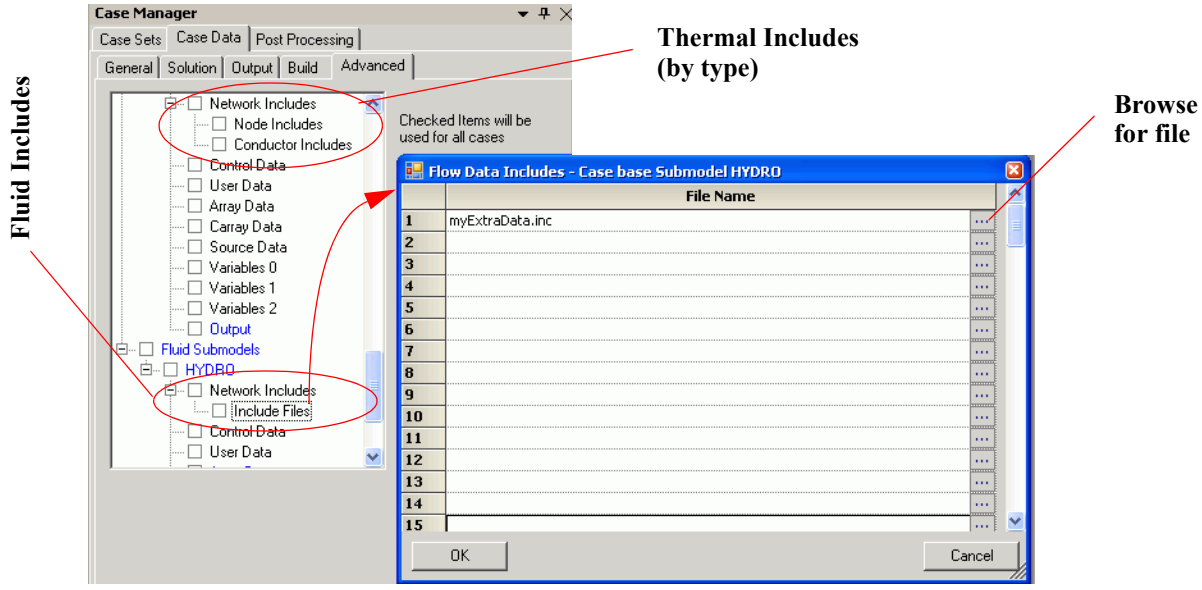
Sometimes, some network elements (nodes, lumps, paths, etc.) need to be added to a model when SINDA/FLUINT is launched, but there is no need to visualize the data nor to edit it. For example, radiation conductors ("RADKs") are sometimes produced by external programs and would result in a dense mesh if depicted on the Network Diagram. Also, sometimes nodes and conductors are generated by external programs (e.g., Thermal Desktop, Excel, etc.), or are from older models that are too large to import, or which cannot be modified.

Network Includes provides a way of browsing for an INCLUDE file to be inserted at run time. This file will not be imported nor even read until SINDA/FLUINT execution starts.

For thermal models, the user must place each set of data to be included into separate files according to both submodel and to element type (node vs. conductor). For fluid submodels, the files must be organized by submodel, but there is no further subdivision required by element type. The files to be imported must match SINDA/FLUINT text input formats for the type of file being imported, *and no "HEADER" statements should be encountered in these files.*

Under each submodel of the Advanced tab's model tree, an opportunity exists to define a submodel-specific Include file. For thermal models, separate files can be browsed for or entered under Network Includes>Node Includes and Network Includes>Conductor Includes. For fluid submodels, all extra lump, path, tie, etc. data for any one particular submodel may be entered in Network Includes>Include Files. An example is shown in Figure 9-22.

FIGURE 9-22. Example of Network Includes



9.3.5.14 Submodel-specific Control Data

Global level control inputs ("Global Control Data" on page 9-19) can be overridden at the submodel level if needed. Figure 9-23 shows a thermal submodel with DRLXCA and ARLXCA being reduced to 0.001, with other values left at the global value via the default "GLOBAL" designation. Similarly, Figure 9-24 shows a fluid submodel with steady state output interval (ITROTF) and transient per-time-step output flag (OPITRF) being overridden while other constants are left as the global value. Both forms have Advanced tabs for more rarely needed control options, and both forms provide explanatory hover text at the bottoms.

FIGURE 9-23. Thermal Submodel-specific Control Data

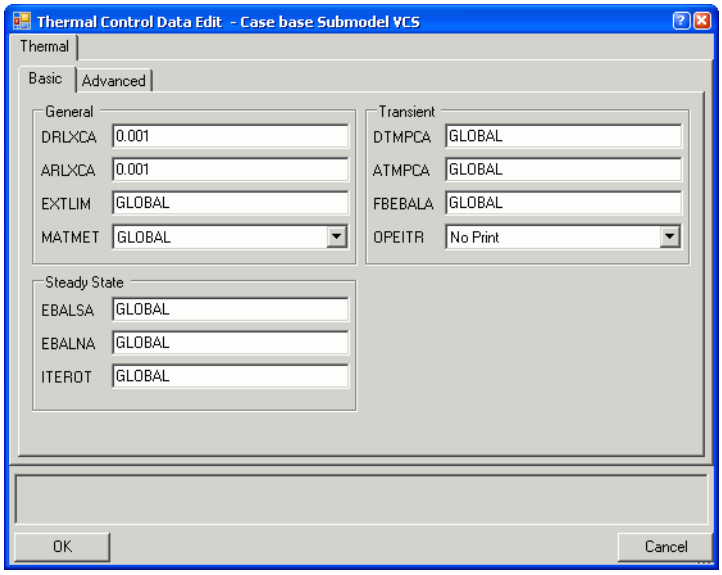
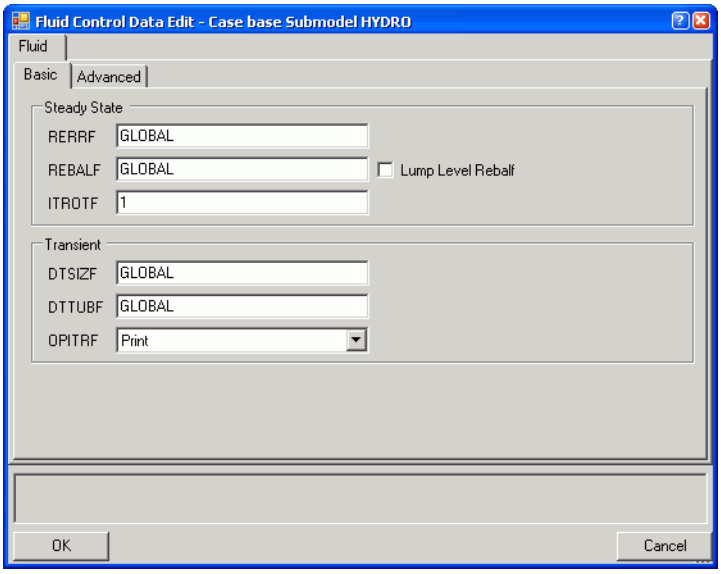


FIGURE 9-24. Fluid Submodel-specific Control Data



9.3.5.15 Carray Data

Carray ("character array") data are special purpose SINDA/FLUINT arrays that contain undelimited strings of characters. Once Carray is allowed per line, imposing a 1000 character limit on any such string. For example, the following line defines Carray #456 in the current submodel.

```
456 = This is an example of a Carray
```

Carrays are rarely used. See Section 2.11.3 in the SINDA/FLUINT User's Manual for details.

9.3.5.16 Source Data

Nodal heat sources can come from many causes: electrical dissipation, planet albedo, aeroheating, etc. For this reason, their definition can be complex, and multiple sources can be defined for any one node, in which case their action is cumulative (and will be summed into the Q array).*

The Source tab on the Node Edit form allows one source to be specified per node. Additional sources can be identified under the Advanced tab of the Case Manager's Case Data tab, in the Source Data section of the options tree. These sources must be located in the correct submodel, and follow SINDA/FLUINT text input format: as with many such options, this is a simple text edit window.

9.3.5.17 User Logic Blocks

In addition to custom output blocks, the user may supply submodel-specific customizations via thermal Variables blocks and fluidic Flogic blocks. Refer to "Submodel-specific Logic Blocks", Section 5.6.6, for details on the meaning of these blocks, including when they are called and how they are typically used.

Submodel-specific logic can be added under the Advanced tab of the Case Manager's Case Data tab, in the Variables, Flogic, or Output section of the options tree. (See "Optional Output Routines" on page 10-4 for examples of custom outputs.) The format follows SINDA/FLUINT and Fortran input format: as with many such options, this is a simple text edit window.

* For this reason, Qs are zeroed at the start of every time step or steady-state relaxation step, and are rebuilt each time. Thus, in logic blocks, the user should always sum into a node's Q (e.g., "Q10 = Q10 + 10") rather than replace it (e.g., *don't use* "Q10 = 10.0" or prior calculated sources on that node might be replaced).

CHAPTER 10

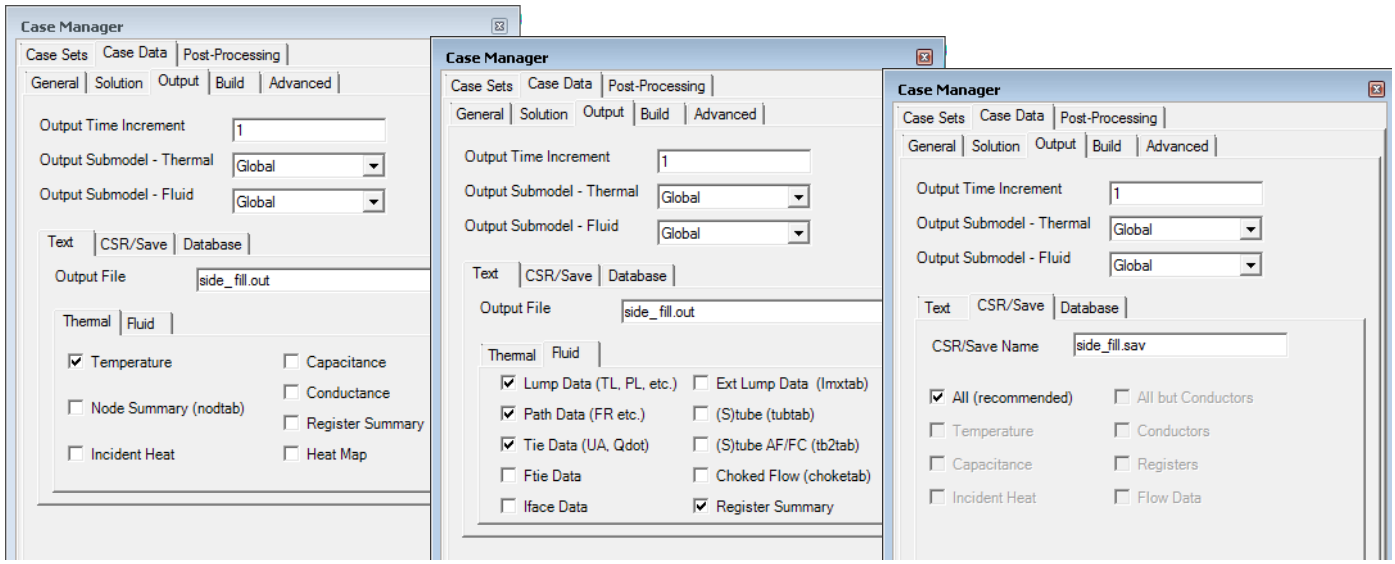
Output and Post-processing

After a case has been executed through the Case Manager (i.e., a SINDA/FLUINT run has been made), the processor results can be examined through formatted text files or visualized through a variety of post-processing options such as coloring the network, thickening links in the network, text listings, and plotting. Post-processing of a model requires the generation of a binary format SAVE file (*.sav), compressed solution results (CSR) folder (*_csr), or a restart file/folder (*.rso). The generation of text output is controlled through Case Manager and is typically in the form of an output file (*.out) and optional user files (*.usr, for example).

10.1 Controlling Output Frequency and Location

The user can customize the output interval and content settings through the Output tab of the Case Manager's Case Data tab, as shown in Figure 10-1. The Text tab applies to the results written to the text-based output file while the CSR/Save tab applies to the binary processor results used for post-processing. By default, common values have been defined for various fields and common output selections are preselected. The logic generated by the contents of this form will be written to the global output block.

FIGURE 10-1. Setting Output Intervals



Output Time Increment - This field controls the frequency of output during a run. The Output Time Increment only applies to transient solutions. This value controls the time period between writes to the output and SAVE files or CSR folders. By default the output increment is set to 1/100th of the total run time (which in turn is determined by the value of the control constant TIMEND). The user can change this data field, however caution should be applied since a large model can generate very large output files.

The user can vary the output interval *during* a transient solution either though user-defined logic (placed perhaps in Variables 2 or Flogic 2) or by defining the interval data field with a conditional operator such as:

```
(timen > 1500)? 0.01*timend : 30
```

meaning if the current time (TIMEN) is greater than 1500 time units, set the output interval for 1/100 of the problem time, else set it to 30 time units

Alternatively they can use the control constants OPEITR (for thermal submodels) or OPITRF (for fluid submodels) to change the output interval to every time step for a portion of the solution. This is often useful when troubleshooting a model, and should be used with extreme caution or some limits, or else excessive output can result.

```
(timen < 850)? 0 : (timen > 900)? 0 : 1
```

meaning if the current time (TIMEN) is less than 850 time units but greater than 900 units, set the write flag to 0 (do *not* write output), otherwise write output every time step within that interval

Similar output control can be used for steady state solutions using the output control constants ITEROT for thermal submodels and ITROTF for fluid submodels. For example, a common debugging trick for debugging nonconvergence is to set either ITEROT or ITROTF to 1 for the last 10 iterations before NLOOPS is reached:

```
(loopct > nloops - 10)? 1 : 0
```

Output Submodel (Thermal/Fluid) - The Output Submodel data fields identify where the output logic will be written. By default, this is set to "Global" which designates the output to be written by the global output block. The user can designate a specific submodel to contain the output logic by using the pull-down menu in the data field. The option "(None)" will result in no output being written. If a submodel is selected instead of the global option, be aware that if the submodel is not built, there will be not output since submodel specific logic is not performed when a submodel is not built.

The Text and Save tabs allow the user to specify the type of output to be written to the text-based and binary save file or CSR folder. These options will be described in the subsections that follow.

The data fields for the Output File Name and the Save File Name are defaulted to [case name].out and [case name].sav or [casename]_csr. These data fields are linked to the associated fields on the General tab of the Case Manager. Changing the names in one form will automatically change them in the other form.

10.2 Text Output Files

Text output files generally contain three types of data: preprocessor output, message and logs associated with the run, and formatted processor output.

The preprocessor output file (*pp.out* by default) contains a listing of the input file as the preprocessor has interpreted it. If there are any errors detected by the preprocessor they will be listed in this file. Search the file for the words "ERROR" and "CAUTION" as needed.

The message and log files (*messages.txt* and *messages.log*, respectively) contain information from the execution of the run. Compilation and link errors are generally listed in the messages.log file while cautions and error messages from the processor are found in the messages.txt file. When a model fails to run properly, the user should inspect the contents of these files. (See "To Run a Case:" on page 9-3.)

Additionally the user can create user-defined output files. These are useful for creating tabulated output tailored to the needs of the analyst. (See "Options Data" on page 15.)

10.2.1 Standard Text Output

The user can control the contents of the standard output file (*.out). A sample of an output file containing a lump tabulation and a path tabulation is shown Figure 10-3, highlighted in red. The header information highlighted in blue contains information regarding the status of the run. If the run fails to converge, this area typically contains information which may aid in trouble-shooting the problem.

When an output option is selected on the Text and Save tabs shown in Figure 10-1, it applies to all submodels. For example if the temperature box is selected on the Thermal tab, the temperatures of all nodes in all *thermal* submodels will be written out. Likewise if an item is selected on the Fluid tab it will apply to all *fluid s*ubmodels. By default, the logic created for all output is placed in the Global Output Calls block. If output is only desired for a specific submodel, custom calls to the output routine(s) should be placed in the appropriate submodel level Output Data logic block (see "Optional Output Routines" on page 10-4).

The various text output options are summarized in Table 10-1.

TABLE 10-1. Text Output Options

Parameter	Description (SINDA/FLUINT Output Routine[a])
Global (listed on both tabs)	
Register Summary	Tabulates registers and expressions (REGTAB)
Thermal	
Temperature	Prints all nodal temperature data (TPRINT)
Node Summary	Tabulates nodal parameters and energy balance data (NODTAB)
Incident heat	Tabulates nodal source data (QPRINT)
Conductances	Prints all conductor values (GPRINT)
Capacitance	Prints all nodal capacitance data (CPRINT)
Register Summary	Tabulates register names and values (REGTAB)
Fluid	
Lump Data	Tabulates basic lump parameters (LMPTAB[b])
Path Data	Tabulates path parameters (PTHTAB[b])
Tie Data	Tabulates tie parameters (TIETAB[b])
Ftie Data	Tabulates ftie parameters (FTTAB)
Iface Data	Tabulates iface parameters (IFCTAB)
Ext Lump Data	Tabulates extra lump data VOL, CX, etc. (LMXTAB)
(S)tube Data	Tabulates tube and STUBE parameters (TUBTAB)
(S)tube AF/FC	Tabulates area and friction data for tubes, STUBEs (TB2TAB)
Choked Flow	Tabulates choked flow parameters (CHOKETAB)
Register Summary	Tabulates register names and values (REGTAB) (For convenience: identical to the thermal option above)

a. See SINDA/FLUINT User's Manual for more information on these and other output routines

b. In case the user forgot to request this data *before* making a run, not that similar data to that produced by these routines is also available *after* a run using the Extended Text option (Section 10.4.5.1).

10.2.2 Optional Output Routines

The most common output options are built into the Output tab of the Case Manager, however additional output routines are available in SINDA/FLUINT. These additional routines are summarized in section 7.4 of the SINDA/FLUINT User's Manual.

To invoke these optional routines, or to create a custom user-defined file, logic must be added in the Output text window which is accessible through the Advanced tab of the Case Manager's Case Data tab, as shown in Figure 10-2. Each submodel has a unique output block allowing the user to customize the output for that submodel. Using the output text window, logic can be added to any output routine. In the sample shown, the call to GPRINT will provide a table of conductor values for every thermal submodel designated (as such by the 'all' parameter in the calling sequence). If the user only wanted conductance data for the thermal submodel *lines*, then the call statement would appear as

```
CALL GPRINT('LINES')
```

By default, this output data will be written to the output file in addition to with any specifications made in the Output tab.

FIGURE 10-2. Creating Custom Text Output

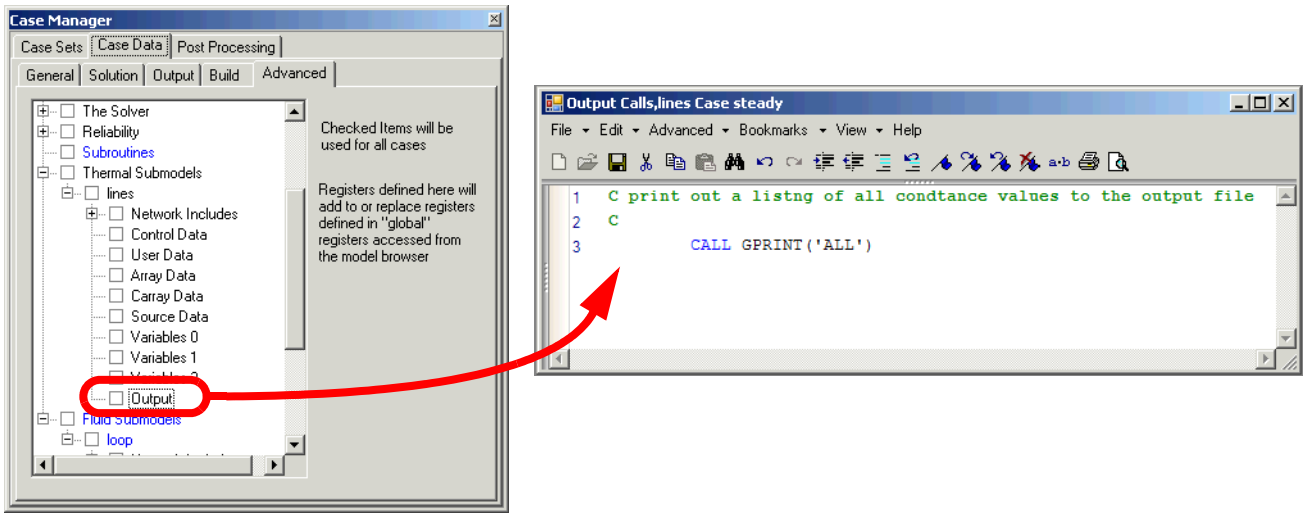


FIGURE 10-3. Tabulated Output File

```
FLUID SUBMODEL NAME = PIPE    ; FLUID NO. =  718

      PROBLEM TIME                  TIMEN  =  1.00000     VS. TIMEND  =  10.0000    ; DTIMES  =  0.900000
      CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF      28 ITERATIONS
```

LUMP PARAMETER TABULATION FOR SUBMODEL PIPE

LUMP	TYPE	TEMP	PRESSURE	QUALITY	VOID FRACT.	DENSITY	ENTHALPY	HEAT RATE	MASS RATE	ENERGY RATE
100	TANK	273.2	604.7	0.8709	1.000	5.5115E-03	2.1757E+06	-45.98	0.000	-8.7280E-03
101	TANK	273.2	604.7	0.8707	1.000	5.5127E-03	2.1753E+06	-45.98	0.000	-3.1471E-03
102	TANK	273.2	604.7	0.8705	1.000	5.5138E-03	2.1748E+06	-45.97	0.000	9.8000E-03
103	TANK	273.2	604.7	0.8703	1.000	5.5150E-03	2.1743E+06	-45.96	0.000	5.4817E-03
104	TANK	273.2	604.7	0.8701	1.000	5.5161E-03	2.1739E+06	-45.95	0.000	-4.4289E-03
105	TANK	273.2	604.7	0.8699	1.000	5.5173E-03	2.1734E+06	-46.01	0.000	1.5602E-03
106	TANK	273.2	604.7	0.8698	1.000	5.5185E-03	2.1730E+06	-45.90	7.4506E-09	2.8080E-02
107	TANK	273.2	604.7	0.8696	1.000	5.5196E-03	2.1725E+06	-45.92	-7.4506E-09	-2.1931E-02
108	TANK	273.2	604.7	0.8694	1.000	5.5208E-03	2.1720E+06	-45.90	0.000	-1.0777E-02
109	TANK	273.2	604.7	0.8692	1.000	5.5219E-03	2.1716E+06	-45.86	0.000	9.1209E-03
2000	JUNC	364.4	604.7	1.000	1.000	3.5974E-03	2.6716E+06	0.000	0.000	-1.0231E-02
2001	PLEN	371.4	9.5000E+04	1.000	1.000	0.5631	2.6739E+06	0.000	0.1000	2.6716E+05
1001	PLEN	372.8	1.0000E+05	1.000	1.000	0.5908	2.6762E+06	0.000	-0.1000	-2.6762E+05

```
                        SYSTEMS IMPROVED NUMERICAL DIFFERENCING ANALYZER WITH FLUID INTEGRATOR          PAGE     5
DATE = 11/06/2006 START TIME = 15:09:55
MODEL = SINDA                                    Pipe Prebuilt
  FASTIC


 FLUID SUBMODEL NAME = PIPE    ; FLUID NO. =  718

       PROBLEM TIME                  TIMEN  =  1.00000     VS. TIMEND  =  10.0000    ; DTIMES  =  0.900000
       CONVERGENCE STATUS = SUBMODEL CONVERGED AS OF      28 ITERATIONS
```

PATH PARAMETER TABULATION FOR SUBMODEL PIPE

PATH	TYPE	LMP 1	LMP 2	DUP I	DUP J	STAT	XL UPSTRM	FLOWRATE	DELTA PRES	REYNOLDS	FR/FRC	REGIME
100	TUBE	1001	100	1.0	1.0	NORM	1.000	0.1000	9.9395E+04	5.2922E+05	1.178x	1 PHASE
101	TUBE	100	101	1.0	1.0	NORM	0.8709	0.1000	0.000	6.8976E+05	9.999	ANNULAR
102	TUBE	101	102	1.0	1.0	NORM	0.8707	0.1000	0.000	6.8962E+05	9.999	ANNULAR
103	TUBE	102	103	1.0	1.0	NORM	0.8705	0.1000	0.000	6.8948E+05	9.999	ANNULAR
104	TUBE	103	104	1.0	1.0	NORM	0.8703	0.1000	0.000	6.8933E+05	9.999	ANNULAR
105	TUBE	104	105	1.0	1.0	NORM	0.8701	0.1000	0.000	6.8919E+05	9.999	ANNULAR
106	TUBE	105	106	1.0	1.0	NORM	0.8699	0.1000	0.000	6.8904E+05	9.999	ANNULAR
107	TUBE	106	107	1.0	1.0	NORM	0.8698	0.1000	0.000	6.8890E+05	9.999	ANNULAR
108	TUBE	107	108	1.0	1.0	NORM	0.8696	0.1000	0.000	6.8876E+05	9.999	ANNULAR
109	TUBE	108	109	1.0	1.0	NORM	0.8694	0.1000	0.000	6.8861E+05	9.999	ANNULAR
110	TUBE	109	2000	1.0	1.0	NORM	0.8692	0.1000	0.000	6.8847E+05	9.999	ANNULAR
2002	MFRSET	2000	2001	1.0	1.0	NORM	1.000	0.1000	-9.4395E+04			

10.2.3 User-defined Output Files

The user can optionally create custom output files by writing data to the *.user1 file or opening additional user files. In order to write to a user file, the file must first be defined in Options Data through the Case Manager (Case Data>Advanced>Options Data). For more information on user files, see section 7.4.16 of the SINDA/FLUINT User's Manual. Fortran write statements can added to any logic block.

Examples of unformatted write statements to a user file or output file:

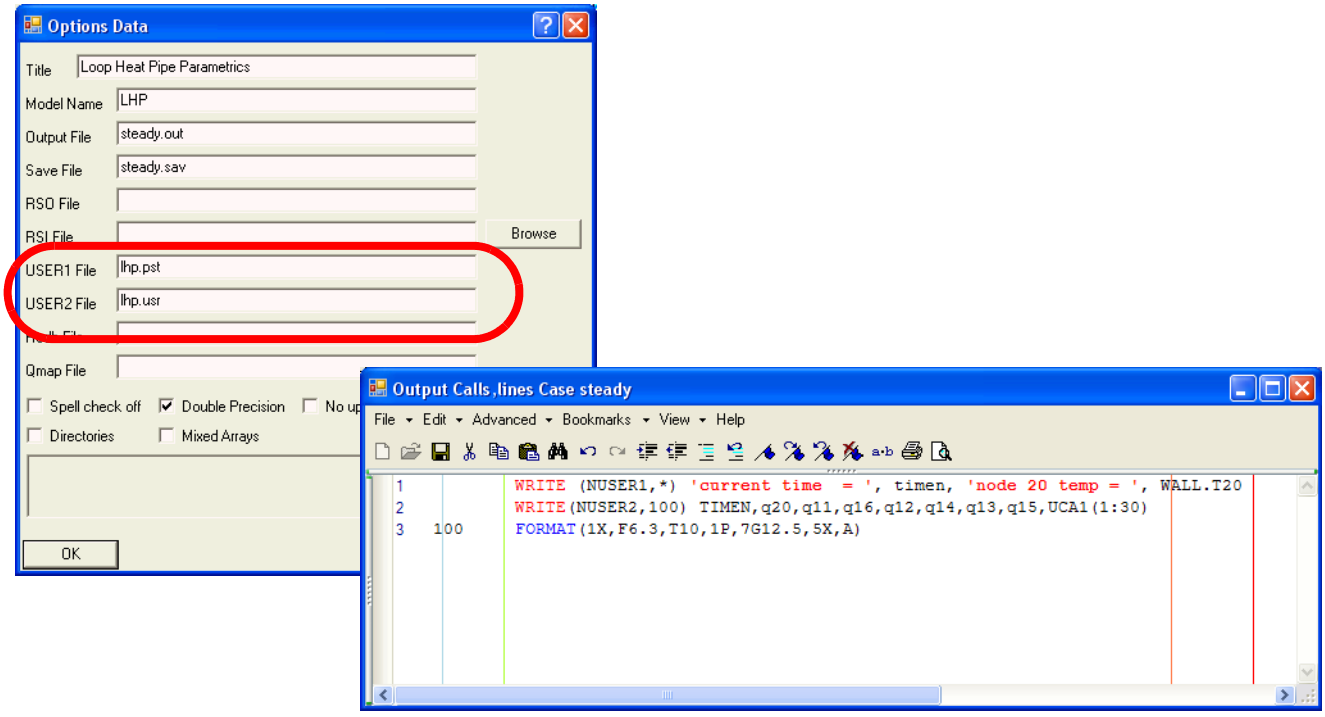
```
WRITE (NUSER1,*)  'current time  = ', timen, 'node 20 temperature = ', WALL.T20

WRITE (NOUT,*)  'current iteration= ', nloops, 'conductor 120 heat rate = ', plate.HR120
```

Examples of a formatted write statements to a user file or output file:

```
      WRITE(NUSER2,100) TIMEN,q20,q11,q16,q12,q14,q13,q15,UCA1(1:30)
100   FORMAT(1X,F6.3,T10,1P,7G12.5,5X,A)
```

Figure 10-4 depicts an example of defining user files through Options Data and associated write statements.

FIGURE 10-4. Creating User-defined Output



10.3 Binary Output for Postprocessing

A relatively small number of output routines produce binary (machine readable) instead of ASCII text. Each binary routine produces content that is directed to a special file, usually unique to that routine. The most common routine is the SAVE routine, which writes to a Compressed Solution Results (CSR) folder or, a SAVE file, as specified in the Case Manager. However, other binary routines exist too, including:

SAVE . . . . . Principal binary output routine used for post-processing, writes to a SAVE file/folder
RESAVE. . . . . For restart purposes, writes to the restart (RSO) file/folder
CRASH. . . . . For crash recovery (see the SINDA/FLUINT Manual for more information)
SAVEDB . . . . . Cumulative saves to the REDB file, read by the RESTDB routine
SAVPAR. . . . . Saves all to an temporary file for resetting within a parametric or Solver run
SVPART . . . . . Selective version of SAVPAR

Note that the format produced by SAVE, RESAVE, CRASH, and SAVEDB are all identical. In other words, files produced for one purpose (e.g., post-processing) can also be used for a different purpose (e.g., restarting). However, note that when restarting a fluid submodel, a minimum of information must have been stored on the binary output.

For more information on the various binary output formats and usage, please see the SINDA/FLUINT User's Manual.

10.3.1 Creating Binary Output (SAVE file or CSR folder)

SAVE files or CSR folders are required for post-processing operations in Sinaps. The name of the binary file/folder is specified on the General tab of the Case Manager in the CSR/Save for Post-processing field (see section "Case Data - General" on page 9-4). The frequency of writing the binary output is controlled similar the text output as described in Section 10.1. The main menu option Model>Use CSR Folder controls whether binary output is saved to a SAVE file or CSR folder).

Each time the SAVE routine is called, it adds a sequential "snapshot" or ***record*** to the binary file/folder. Thus, if it is being called during a transient, it is building up a history of responses versus time. If it is called during a parametric sweep, it is storing responses as a function of the swept parameter.

Usually, the content of the binary output records contain "*all*" parameters associated with every network element along with register information, control constants, and arrays. The user can tailor the contents of the output by using the check box options on the CSR/Save tab on the Case Data>Ouput tab in the Case Manager as shown in Figure 10-5. The format for user-supplied calls is as follows:

```
CALL SAVE ('ARGS',NFLAG)
```

with a typical call as follows:

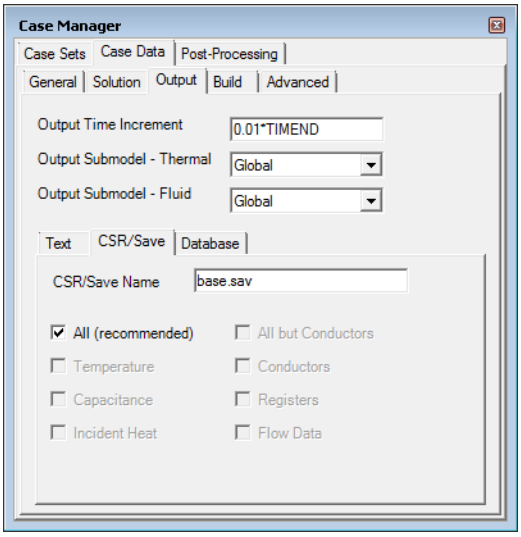
```
CALL SAVE('ALL', 0)
```

> Note: ARGS must always be enclosed in single quote character delineators ('). In order to restart FLUINT models, ARGS must either be 'ALL' or include 'LPFM'.

The options for ARGS are summarized in Figure 10-5. The "All" column indicates what is saved if the default option "All" is selected. The "ARGS" column identifies the various argument available for the calling sequence if the user wishes to create a custom (reduced content) call to the SAVE output routine. Any custom call to the routine can be placed in Operations or in any output logic block.

NFLAG controls whether or not the current values of the control constants are printed when SAVE is called. If NFLAG = 1, the control constants and model energy balance data will be printed to the output file. If NFLAG = O, only the message "SAVE called at TIMEN = xxxxx" will appear.

FIGURE 10-5. Controlling Binary Output Content



Option	Description	"All"	ARGS
Temperature -	Write temperatures	yes	T
Capacitance -	Write capacitance values	yes	C
Incident Heat -	Write Q-source values	yes	Q
Conductors -	Write conductor values (G and HR)	yes	G
	Write control constants	yes	N
	Write *numbered* user constants	yes	U
	Write user arrays and CARRAYs	yes	A
Flow Data -	Write lump PL, TL, XL, and QDOT	yes	L
	Write path FR to SAVE (for plots)	yes	P
	Write advanced mixture and iface data	yes	M
	Write remaining FLUINT data	yes	F
Registers -	Write register names and expressions	yes	R

10.4 Post-processing Operations

Post-processing options include coloring nodes and lumps in a network, thickening links in a network, plotting results, animating results, viewing hover text, and generating node and lump balance plots and extended text output tables. When post-processing the results, the user must identify:

1. which binary output (SAVE or CSR) to use, and perhaps what time or saved point within that file
2. what data to report (e.g., temperatures, pressures, flow rates)
3. which network elements are to be post-processed, and
4. what type of post-processing is to be performed (e.g., balance plots, color, plot vs. time, animate)

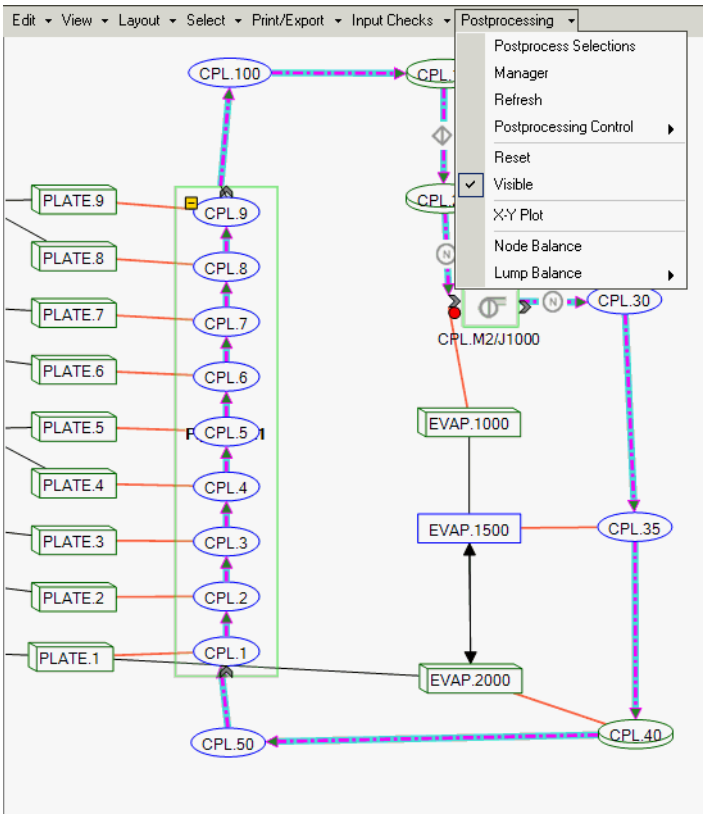
These questions are answered by Case Manager choices, Network Diagram element selections, and via the options available through the Post-processing pull-down menu in the Network Diagram as shown in Figure 10-6. The CSR/Save Manager button on the Case Manger Post-Processing tab is used for binary output file/folder control and for data type selection, while the Post-processing Control menu option is used for time step control and animation. Basically, the Case Manager's CSR/Save Manager tells the program *where to search for the results* (SAVE files or CSR folders, and perhaps location or record within those files), while the Post-processing tab tells the program *what type of data* to present. Element selections in the Network Diagram tell the program *which objects* to color or plot. Finally, the Post-processing pull-down tells the program *what to do* (plot, color, thicken, etc.) based on that results data.

> Tip: Because the Case Manager plays an important role in post-processing, that window should be kept visible, either adjacent to the Network Diagram or floating above it.

By default, Sinaps will automatically set the binary output file/folder information and post-process the entire model (color and/or thicken the network) at the completion of a run based on the post-processing controls set in the Post-processing tab of the Case Manager.

The remainder of this section describes how to extract more specific data that might required, in the desired presentation format.

FIGURE 10-6. Post-processing Options



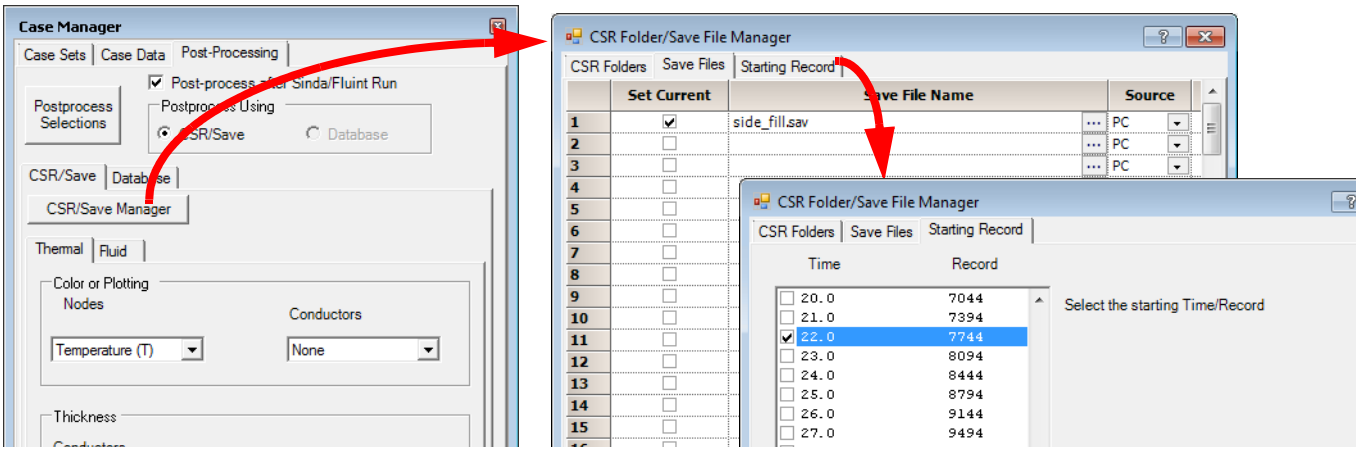
10.4.1 Case Manager Post-processing Tab

The CSR/Save File Manager is accessible from the Case Manager's Post-processing tab as shown on the left side of Figure 10-7. This dialog allows the user to change or binary postprocessing files or folders in addition to controlling which records (or times) are to be used for post-processing the network. By default Sinaps automatically sets the source file based on data in the Case Manager for the current case.

When the CSR/Save File Manager button is pressed, the CSR FOlder/Save File Manager dialog will appear. This window has three tabs: one for setting the CSR Folder to use for post-processing; one for setting the SAVE file (if used instead of a CSR Folder); and one for selecting the time/record within that data source to use for current network post-processing, or the starting point for a transient plot or text post-postprocessing. The options on each of these tabs are the subject of the Section 10.4.1.1.

The bottom portion of the Case Manager's Post-processing tab, further described in Section 10.4.1.2, controls what type of data should be displayed when plotting, coloring, etc. For example, when a lump is selected, which of its values should be shown or plotted: temperature, pressure, volume, species concentration, or some other value?

FIGURE 10-7. Post-processing Manager and Save File Control



10.4.1.1 CSR Folder/Save File Manager

The CSR/Save File Manager button (Figure 10-7) will opens the CSR FOlder/Save File Manager dialog. This dialog can be used to manage which source files are to be used during post-processing operations.

CSR Folders: The CSR Folder tab provides a list of CSR folders to be used as the source for post-processing. Only a single file can be post-processed at a time, but multiple files can reside in the list. Each file might represent a the same results produced using a different parameter or control setting, variations in logic, etc.

The default CSR folder is the one defined on the General tab of the Case Manager. dialog. When a model has been run and automatically post-processed, the default CSR folder is used for post-processing. The CSR Folder dialog allows the user to *replace* the default folder by a different folder, or to browse for *additional* CSR Folders. The user can specify several CSR folders to be associated with a model. The series of Set Current check boxes in the left column designate which folder is currently active for network post-processing and plotting. *Only folder can be active at a time.*

Using the Source column, the user can define the source machine from the available machine types: PC (default) or UNIX. Use the pull-down options to select the type of machine that produced the CSR folder (e.g., if you are given a applicable folder from co-worker which was created on an UNIX based machine - select UNIX).

> Note: The CSR Folder must correspond to the current model network executed under a compatible version of SINDA/FLUINT. Otherwise, an error may occur during post-process-

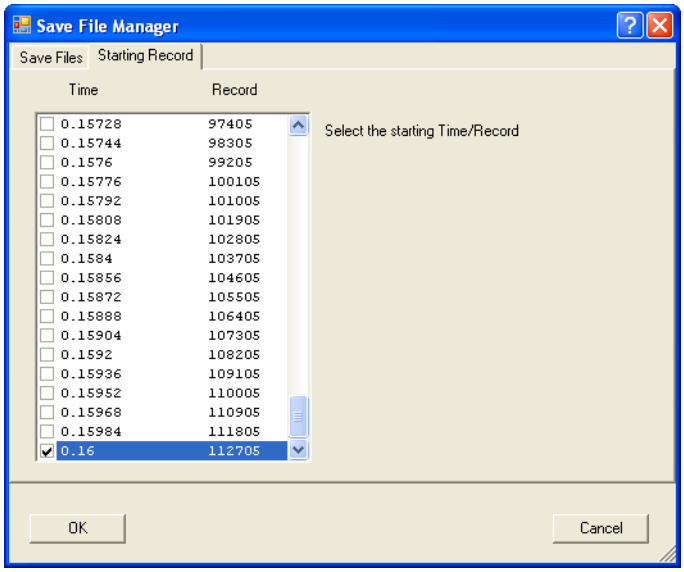
Save Files: The Save File tab provides a list of SAVE files to be used as the source of results for post-processing. Only a single file will be post-processed at a time, but multiple files can reside in the list. Each file might represent a the same results produced using a different parameter or control setting, variations in logic, etc.

The default SAVE file is the one defined on the General tab of the Case Manager. dialog. When a model is run and automatically post-processed, the default save file is used for post-processing. The Save File Manager dialog allows the user to *replace* the default file by a different file, or to browse for *additional* SAVE files. The user can specify several SAVE files to be associated with a model. The series of Set Current check boxes in the left column designate which SAVE file is currently active for network post-processing and plotting. *Only one save file can be active at a time.*

Using the Source column, the user can define the source machine from the available machine types: PC (default) or UNIX. Use the pull-down options to select the type of machine that produced the save file (e.g., if you are given a applicable save file from co-worker which was created on an UNIX based machine - select UNIX).

> Note: The SAVE file must correspond to the current model executed under a compatible version of SINDA/FLUINT. Otherwise, an error may occur during post-processing.

Starting Record: The Starting Record tab shown below allows the user to select a specific time/record for network postprocessing. The Time column lists the time of the various output intervals and the Record column lists the associated record numbers for each output interval. The user-selectable check box designates which record is active for post-processing. For network post-processing operations such as coloring or thickening of the network, or flow/energy balance plots, only a single record can be selected. By default, network post-processing is always based on the last time or record in the list, and XY plots cover the full time spectrum regardless of which record is checked.



> Note: ***A record number*** is a numeric identifier generated automatically by SINDA/FLUINT every time data is written to a *.sav or *.rso file via the routines SAVE, RESAVE, CRASH, or SAVEDB. Besides postprocessing, a record number can be used for restart operations, see "Options Data" on page 9-15.

For steady state cases, by default only two records will appear in the source file/folder: one for initial conditions* and the second for the final results.

For parametric runs in which time does not vary, the user must use the record number alone: time values in the Starting Record table will not be unique (and are often all zero). In such a case, to help associate record numbers with the correct output, the record number can be found in the output file (*.out) as shown in the example below. In this situation it is beneficial to include a Register Summary in the text output to provide a means of association between the parametric values and the record numbers.

```
SAVE WRITTEN AT TIMEN =   0.00000    , LOOPCT/O/R =  0/  0/  0, STARTING RECORD =      10883
```

10.4.1.2 Selecting Data to Post-process

The user has full control over which parameters are post-processed (plotted, colored, etc.) for the various network elements. These controls are accessible through the Post-processing tab of the Case Manager as shown in Figure 10-8.

The Thermal tab provides options to set the post-processing of thermal submodel elements, while the Fluid tab provides options for fluid submodel post-processing. Both these tabbed forms provide options for coloring (network objects and links) and thickening (links only) through various pull-down menus of parameter selection windows. For example, a network could be colored for thermal nodes and conductor heat rate values, thickening conductors based on the temperature drop across the conductor (delta T) at the same time.

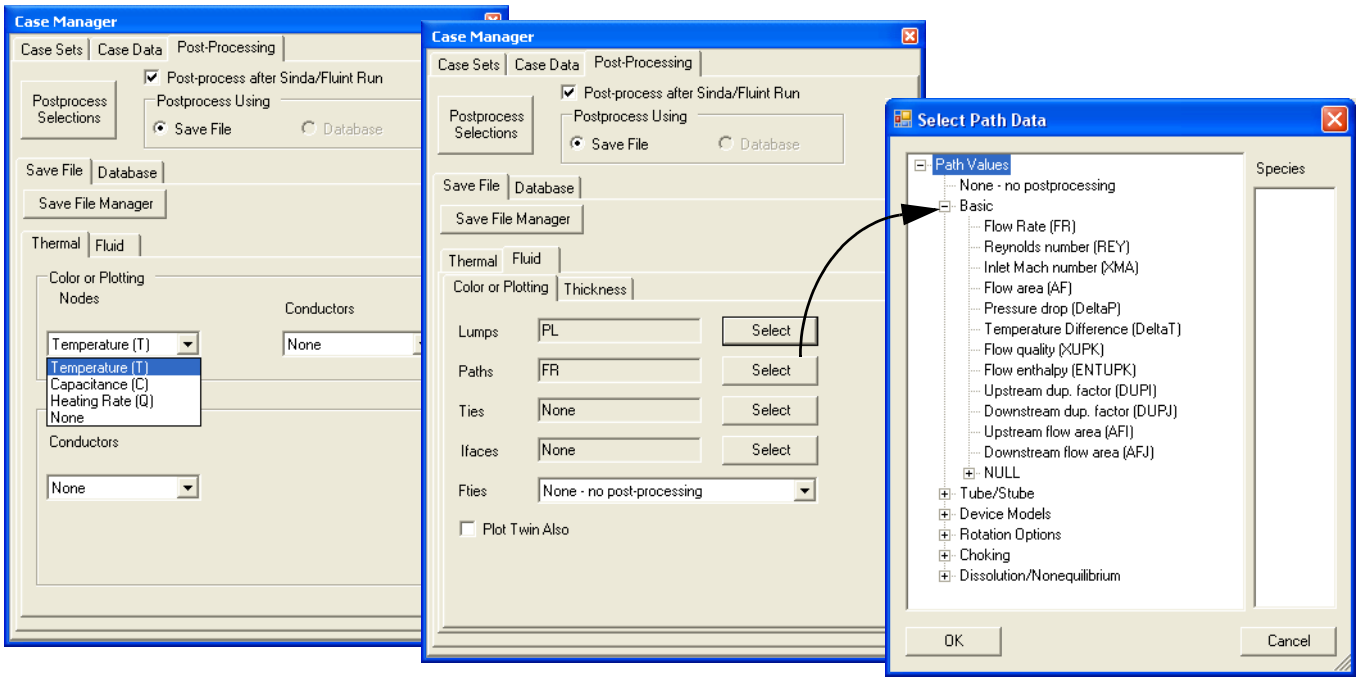
The selected values will also be those used for hover text and XY plots, etc.

> Tip: To navigate the trees quickly, type in the first letter of the desired option. The highlighting will skip to the next parameter description (of those that are visible) that starts

* Transients are often performed using the results of a prior steady state solution. When plotting transient results, the fact that the initial (often guessed) state of the network also appears on the time-history plot can be annoying. Such an initial state is saved by the default first save of the steady state run. Therefore, to avoid having to manually exclude this first record from plots (by adjusting X axis scales, for example), consider adding the following line as the *first* line in the Output block for the relevant submodels:

```
      if(loopct .eq. 0) return          $ don't save guessed initial state
```

FIGURE 10-8. Selecting Values to Post-process



For nodes and conductors, the choices of values to plot or color are few. For fluid submodels, there are hundreds of possibilities as can be seen in Figure 10-9. For each type of element (e.g., lump), there will be choices at the next level (type of tank, type of phenomenon, etc.) in an expanding tree of possibilities. These choices are too numerous to document individually, and the underlying SINDA/FLUINT parameter name is listed in the trees in parentheses. Therefore, any unfamiliar parameter can be searched in the SINDA/FLUINT User's Manual to provide details on its usage.

For fluid mixture parameters that also require the specification of a constituent letter ID, the pane at the right of the form will be populated with possible options, as depicted in the Select Lump Data form at the left of Figure 10-8.

A few parameters are derived: they aren't based on SINDA/FLUINT variables but are largely self-explanatory, such as "Temperature Difference (DeltaT)." A few that aren't as obvious, and that don't correspond directly to a SINDA/FLUINT variable, are listed below. While most of parameters can be used for coloring the network, plotting, or sending as text to Excel, derived values are only available for coloring the network.

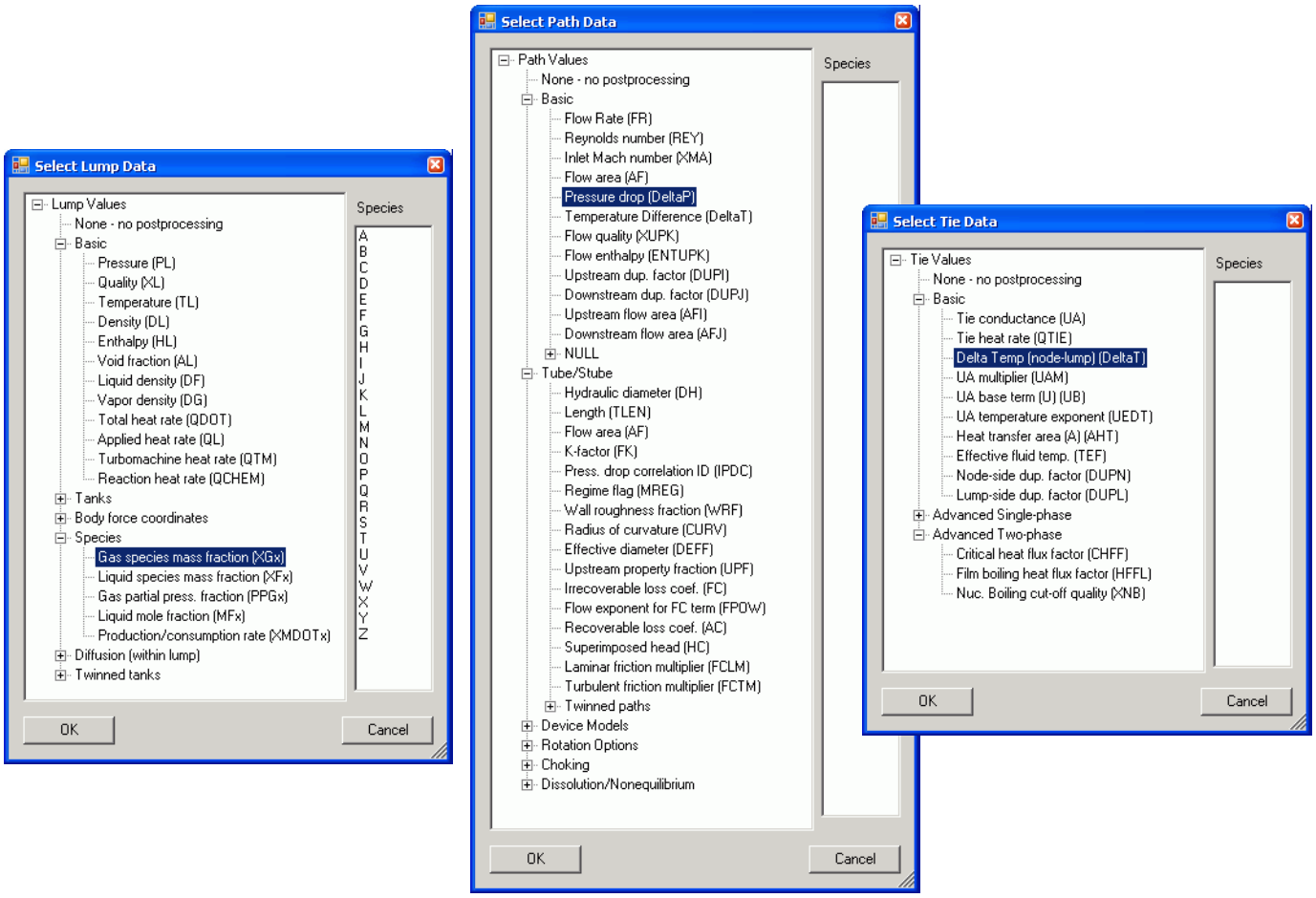
Lumps:

Extensive internal energy (Energy). Internal energy is enthalpy minus pressure divided by density: $u = h - p/\rho$. *Intensive* means per unit mass, while *extensive* means absolute energy: intensive times tank mass: $U = u*M$.

Paths:

Flow quality (XUPK). This is the quality flowing into the path at the current upstream end.
Flow enthalpy (ENTUPK). This is the enthalpy flowing into the path at the current upstream end.
Regime flag (MREG). This is an integer flag to allow the current (inlet) two-phase flow regime to be visualized or plotted: 1=bubbly, 2=slug, 3=annular, and 4=stratified.

FIGURE 10-9. Extensive Parameter Selection Menus Based on Type



10.4.2 Coloring and Thickening a Network Diagram

Once a model has been run, a network can be colored and thickened by selecting the menu option Post-processing>Post-process Selections in the network window. The values displayed will be the default ones (namely, node and lump temperature), with other choices possible (as explained in Section 10.4.1.2). If two or more network elements are selected prior to selecting this menu option, only the selected objects will be post-processed. Note that multi-conductor elements cannot be post processed in the network sine they represent more than one element.

Additional post-processing options accessible through the Post-processing>Post-processing Control menu provide access to stepping through time using Next Time, Previous Time, Current Time, in addition to animating through time via the Animate option.

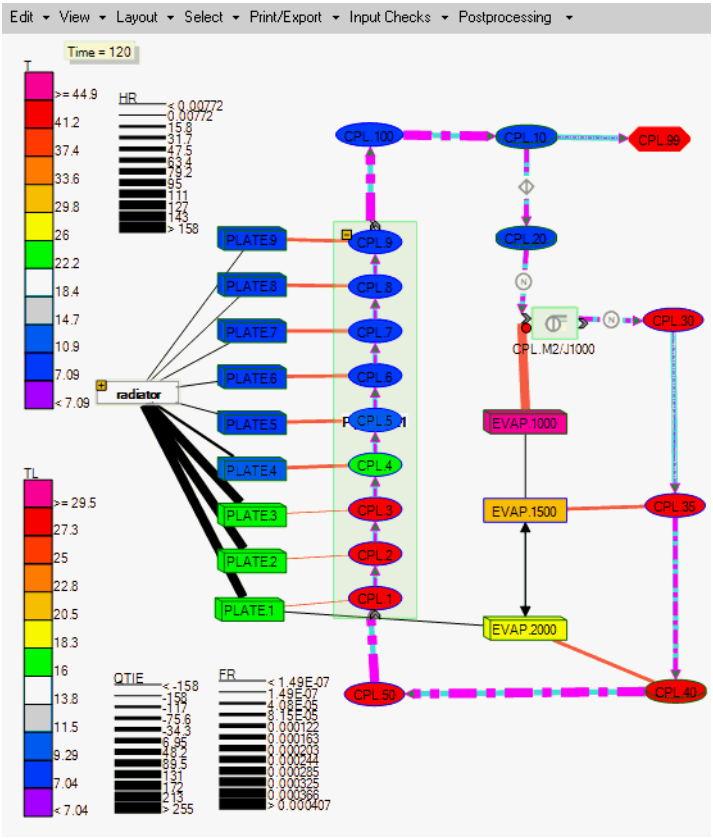
> Tip: Before animating, turn off autoscaling for all color bars and thickness scales, and set the limits to the expected range of variation. Otherwise, the scales change but the dia-

Reset turns off *and resets* all post-processing, whereas Visible is a toggle which turns off the visibility of the post-processing *but retains all customization information* (such as color bar locations).

Figure 10-10 shows an example of a post-processed network. The thermal nodes have been colored based on temperature (T) while the conductors have be thickened based on absolute value of heat rate through the conductors (HR). The fluid lumps have likewise been colored based on temperature (TL) while the paths are thickened based on flow rate (FR), and the heat transfer ties have been thickened based on heat rate through the tie (QTIE, positive from node to lump). Pipe macros and collections (such as the Radiator collection in this figure) are also post-processed, however the post-processing is only visible in the expanded view of the subnetwork.

Sinaps will provide a separate color scale or thickening scale for each network element parameter (node temperature, lump temperature, flow rate, etc.) that is post-processed. These scales are customizable through the Object Properties task panel (main Sinaps menu View>Properties).

FIGURE 10-10. Sample of a Post-processed Network



When a network is post-processed, all the scales are initially located in the upper left corner of the network window. The user can optimize the location of the various scales by moving them. Color scales can also be rotated to be horizontal using the Object Property panel. Once the layout is arranged to the preference of the user, this custom layout will also be saved the next time the model is saved. The layout can be reset to the default layout by clicking on the Post-processing>Reset menu item in the network window, *but otherwise* Visible *should be used instead to retain the layout customizations.*

Relocating Scales: To move a scale left click on a scale, and while holding the mouse button down drag it to a new location.

Synchronizing Scales: Common scales can be synchronized such as node and lump temperatures, or heat rates through conductors and ties. To synchronize scales, select (highlight) two scales and right-click the mouse to select Synchronize scales. When synchronized, the two scales will be expanded to cover the entire scale of each. Specifically, synchronizing does the following:

1. turns off autoscaling for both scales
2. sets the maximum value for both to be the current maximum of both scales
3. sets the minimum value for both to be the current minimum of both scales

Thus by synchronizing the lump and node temperature scales, all objects will be colored based on the same scale, allowing them to be visually compared.

The unsynchronize option simply turns off the autoscaling option. Once the scales have been unsynchronized, the diagram can be post-processed and the scales will scale independently.

10.4.2.1 Customizing Post-processing Scales

Post-processing scales are customizable through the Object Properties task panel. The names of the scales can be changed, their orientation can be changed from vertical to horizontal orientation (color bars only), the resolution can be increased or decreased, the values displayed can be changed to absolute values, or the scale format for numbering can changed.

Autoscale: By default the range of the scale is automatically determined. The user can opt to disable autoscaling by setting the Autoscale value to "false." When set to false, addition fields for Max Value and Min Value will appear in the Object Properties dialog. Subsequent post-processing of the model will use the min/max scale values. This can result in data being out of range if the scale is too restrictive. If this happens, simply edit the scale and reset autoscale to true.

Divisions: The resolution of the color bars and thickness scales defaults to 10: this is the number of colors or thicknesses used. Increasing this number will also increase the size of the scale accordingly.

Scale Format Options: The numbers on the scales corresponding to each color or thickness can be customized using a formatting option. A standard numeric format string takes the form Axx, where A is an alphabetic character called the ***format specifier***, and xx is an optional integer called the ***precision specifier***. The precision specifier ranges

from 0 to 99 and affects the number of digits displayed. Table 11 describes the standard numeric format specifiers. For example, the format G3 refers to General format using 3 significant digits output.

TABLE 11. Standard Numeric Format Strings

Format Specifier	Name	Description
G or g	General	The number is converted to the most compact of either fixed-point or scientific notation, depending on the type of the number and whether a precision specifier is present.
E or e	Scientific (exponential)	The number is converted to a string of the form “-d.ddd…E+ddd” or “-d.ddd…e+ddd”, where each 'd' indicates a digit (0-9). One digit always precedes the decimal point. The precision specifier indicates the desired number of digits *after* the decimal point. The *case* of the format specifier indicates whether to prefix the exponent with an 'E' or an 'e'. The exponent always consists of a plus or minus sign and a minimum of three digits.
F or f	Fixed-point	The number is converted to a string of the form “-ddd.ddd…” where each 'd' indicates a digit (0-9). The precision specifier indicates the desired number of decimal places.
N or n	Number	The number is converted to a string of the form “-d,ddd,ddd.ddd…”, where '-' indicates a negative number symbol if required, 'd' indicates a digit (0-9), ',' indicates a thousand separator between number groups, and '.' indicates a decimal point symbol. The precision specifier indicates the desired number of decimal places.
P or p	Percent	The number is converted to a string that represents a percent. The converted number is multiplied by 100 in order to be presented as a percentage. The precision specifier indicates the desired number of decimal places.
R or r	Round trip	This format is supported only for the Single and Double precision types. The round-trip specifier guarantees that a numeric value converted to a string will be parsed back into the same numeric value.

Max Value: Applicable only if Autoscale is set to false. Defines the maximum value for the range of the scale.

Man Value: Applicable only if Autoscale is set to false. Defines the minimum value for the range of the scale.

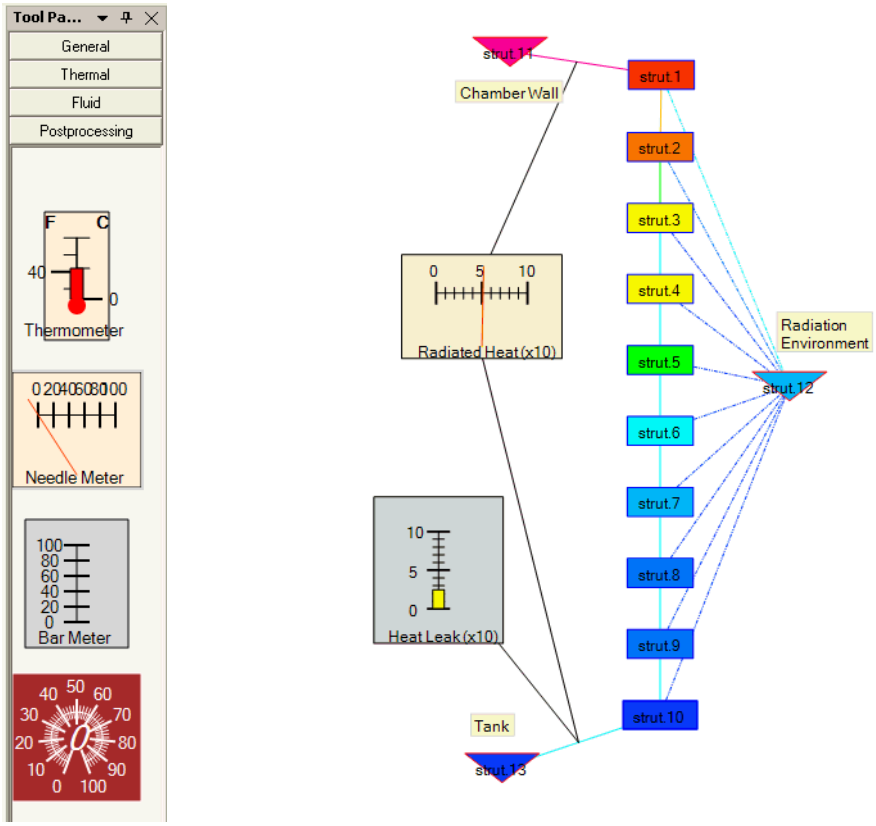
Orientation: Color scales only.) The color scale can be set to display vertically (default) or horizontally.

Title: This is the title for the scale. By default, the title is set as the parameter being post-processed.

10.4.3 Post-processing Tools: Meters, Gauges and Thermometers

Sinaps provides a set of meters and gauges for use in model post-processing. These tools are only available through the tool pallet (View>Tool Pallet), see Figure 10-11. Meters and gauges can be used to display any parameter associated with a network element while a thermometer will only display temperature. Post-processing tools have a *function* option which can be used to display a sum, delta, average, minimum or maximum of multiple (like) elements. They also have adjustable scales and labels.

FIGURE 10-11. Post-processing with Meters, Gauges, and Thermometers



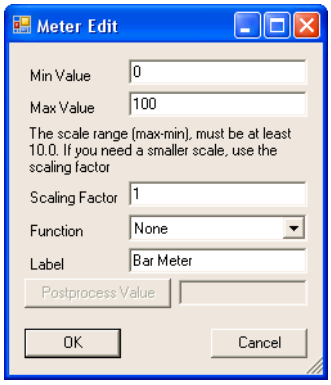
10.4.3.1 Creating a Post-processing Tool

If the tool pallet is not already open, open it through the main Sinaps menu View>Tool Pallet. Expand the Post-processing tab on the tool pallet and click and drag the desired post-processing tool to the network. Alternatively, once selected in the tool pallet, clicking the left mouse button in the network will drop a tool onto the network. Once placed on the network, connect the tool to the network elements by drawing a link to a element. Links can be drawn between the post-processing tool and any node, conductor, lump, path, tie, iface or ftie. Thermometers, on the other hand, can only be connected to either nodes or lumps. Meters and gauges can be attached to more than one network element with the restriction that all connected elements must be the same type of basic element (e.g., all lumps, or all ties, etc.). When a post-processing tool is connected to a mulit-conductor element, it is equivalent to connecting to all the conductors so the function feature will be enabled allowing the summing, averaging, etc. of values. After creating a post-processing tool, it must be edited to specify display data as explained next.

10.4.3.2 Editing a Post-processing Tool

To edit a post-processing tool, double-click on the tool in the network to bring up the edit dialog shown on the left side of Figure 10-12 (form will vary depending on the type of tool being edited). This form provides access to key parameters such as min/max values, labels and more.

FIGURE 10-12. Edit Options for Post-processing Tools



Min/Max Value: The minimum and maximum values for the scale. The smallest range for any postprocessing tool is 10. The scaling factor may be used to meet this requirement. Thermometers only display information in degrees Fahrenheit and Celsius. When defining the min/max values for a thermometers, use the model units. Sinaps will automatically convert the temperature scale for display.

Scaling Factor: Scaling factor for the post-processing data. Sinaps will attempt to autoscale the meter/gauge when it is created. Once created, the user can edit the gauge to set a different scale if desired. This scaling factor is limited to a multiplication factor: expressions cannot be used in this field. This scaling factor can be used to meet the range restriction of 10 between minimum and maximum values on the scale. For example, if displaying flow rate in kg/sec and the values are on the order of 1e-2, set the factor to 1000 to display grams/sec and adjust the range accordingly.

Function: When the tool is connected to more that one network element, a function can be selected using the pull-down menu in the data field. The functions applicable to multiple elements are sum, delta (difference), average, minimum, and maximum.

Label: The label to be displayed on the network with the tool.

Post-process Value: Click the Post-process Value button to select the parameter to be displayed by the tool. In some cases this will open a parameter tree structured list. Thermometers are unique in that they can *only* display temperatures. Thermometers are designed to display in degrees Fahrenheit and Celsius only. If the model is defined using a different temperature scale (i.e., degrees Kelvin or Rankine), the value displayed will automatically be converted into Fahrenheit and Celsius.

10.4.3.3 Customizing the Scales for Post-processing Tools

The scale of a post-processing tool can be customized through the Object Properties dialog, providing the ability to customize tick markings and orientation of the tool (vertical or horizontal), along with duplicating some of the functionality of the Tool Edit form.

LabelDistance: Controls the distance between the tick marks and the tick labels.

LabelFormat: Controls the numeric format for the tick labels. The table “Standard Numeric Format Strings” on page 10-18 provides a formatting key.

LabelFrequency: Controls the frequency of labels for ticks. This setting defaults to 1 for labels on every major tick, 2 for every other major tick.

LabelStyle: Controls tick label justification.

Min/Max Value: The minimum and maximum values for the scale. The smallest range possible between the minimum value and the maximum value is 10; use the *Scaling Factor* to accommodate this limitation as described in Section 10.4.3.2.

Orientation: Toggle between vertical and horizontal layout. This option does not apply to gauges or thermometers.

TickMajorFrequency: Quantity of minor tick marks between major tick marks.

TickUnit: The value between tick minor marks. Note the visibility of the minor tick marks is dependent on the size of the meter.

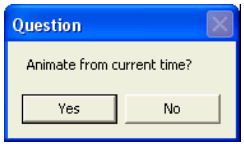
10.4.4 Animation

The model results can be animated through time (transient results) or records (perhaps from a parametric run). To animate the network use the network menu option Post-processing>Post-processing control>Animate. Alternatively the animate icon ( ) can be clicked.

By default when the network is animated, the color bars and network will change based on autoscaling. If the limits of the scales during the animation are known, the autoscaling can be turned off for the post-processing scales and any subsequent animation will result in just the network updating during the animation.

When an animation is started, Sinaps will ask if the animation should start from the current time. By default when a network is post-processor after running, the current time is set for the last record in the save file. To start at the beginning of the save file, reply with “No” to the prompt. To continue from the current record, reply “Yes.” An animation can be paused at any time by clicking the right mouse button.

FIGURE 10-13. Animation Dialog



10.4.5 Text (Excel) Post-processing

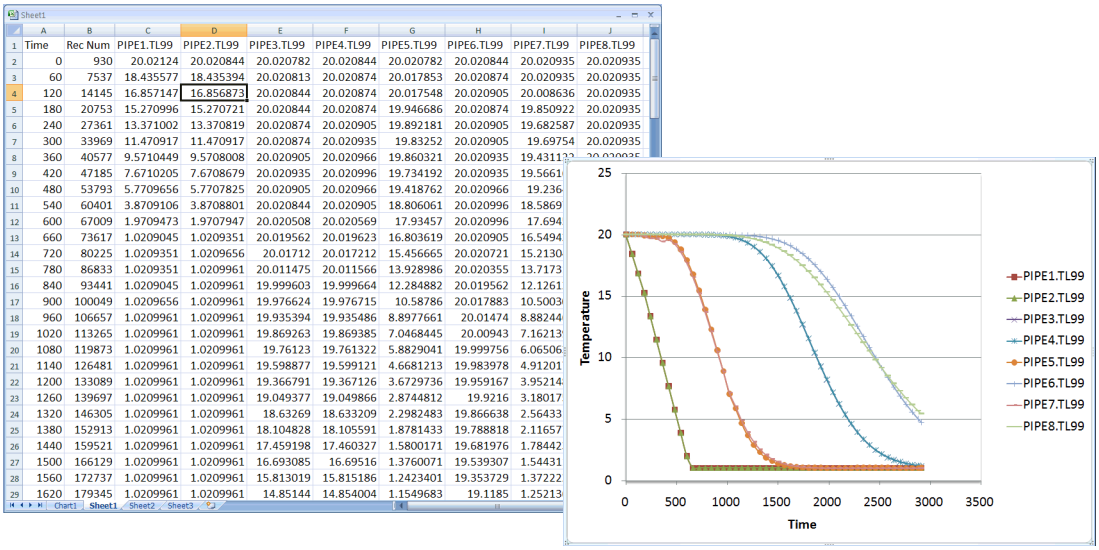
Sinaps provides an option to post-process in a text format. This option sends selected text information from the CSR Folder or SAVE file to a Microsoft Excel™ spreadsheet for inspection, plotting, and perhaps export to another plotting package.*

To post-process for text, select the desired network elements, then select Post-processing>Text from the Network Diagram pull-down menu. Similar to plotting, the text data will be retrieved based on Case Manager post-processing selections for color. If a link is set with thickening but not coloring, no data will be exported for that link.

After selecting Post-processing>Text, an Excel window will open with tabulated data. The resulting data, shown in Figure 10-14, will be displayed as a function of time (if appropriate) and record number (in case time is not appropriate, as is sometimes the case for parametric runs). Once exported to Excel in this fashion, the data can easily be plotted or used elsewhere in Excel. After exporting text to Excel, repeated text post-processing will result in additional worksheets being added to the Excel workbook.

> Note: When the Excel workbook is closed, the connection to the save file is lost. If the user is interested in creating an Excel workbook capable of retaining links to the save file, an Excel Plotter option installed with SINDA/FLUINT is available.

FIGURE 10-14. Example of Text Post-processing in Excel



10.4.5.1 Extended Text

The options under Post-processing>Extended Text are similar to the above options in that they open an Excel window containing tabulated data. However, instead of populating the spreadsheet with the data denoted in the Case Manager

* An Excel-based SAVE file reader is also available as part of the SINDA/FLUINT install set, as is an Excel-based run controller.

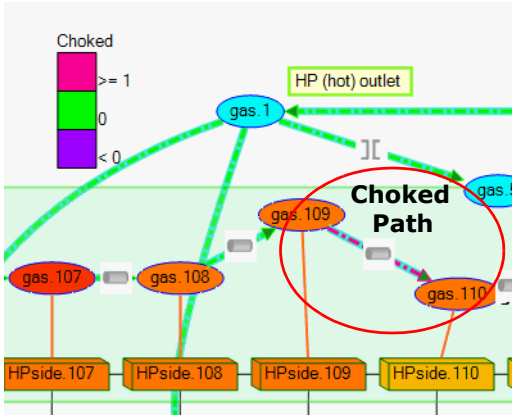
(e.g., “TL” or “FR”), a fixed set of options are available that roughly correspond to the LMPTAB, PTHTAB, and TIETAB tabulations (“Lump Data,” “Path Data,” and “Tie Data,” respectively).

These options are referred to as *extended* because Sinaps is performing some calculations on and interpretations of the data. For example, the “Path Data” option contains pressure drop, regime data, and an indication of whether or not the particular path is current choked. Once a tabulation has been created, the Post-processing>Extended Text->Next Time/Rec and Post-processing>Extended Text->Next Time/Rec options may be used to navigate back and forth through the SAVE file.

Each new request (including the same data at a new time point) is placed on a new Excel sheet.

10.4.5.2 Choked Flow

Fluid networks can be post-processed to visually display choked flow. Using the case manager post-processing tab select the path post-processing option Choking>Choked Flow. Subsequent post-processing of the network diagram will display paths colored based on their choking status. Paths that show a value of 1 are choked while 0 is not choked. Negative values indicate the choking state cannot be determined due to lack of data on the save file. XY plots may also be created based on the choking status.



The choking status of paths can also be determined through the network post-processing menu option Extended Text>Path Data, Section 10.4.5.1, or through the case manager output option Text>Fluid>Choked Flow (CHOKETAB).

10.4.6 XY Plots

An XY plot can be created using the co-installed (but separate) program EZXY®. To create a plot, select the desired network elements, then select Post-processing>X-Y Plot from the Network Diagram pull-down menu. Similar to text export, the plot data will be retrieved based on Case Manager post-processing selections for color. Only parameters used for coloring the network can be used for plotting, i.e. if a link is set with thickening but not coloring, no data will be plotted for that link.

By default, the plot will be generated with time as the X axis. For parametric sweeps and other types of plots, the X axis can be changed after the plot has been generated.

If multiple items are selected, they will all appear on the same XY plot.

> Tip: *Before* calling for a plot, make sure the desired values (e.g., temperature, flow rate, etc.) have been selected in the Case Manager, and select only similar element types (e.g., only lumps, or only paths). This will prevent having to perform as many customizations after the plot has been generated.

For further information on the usage of the EZXY Plotter and the customization of charts, please see the EZXY Users' Manual and recorded webinars on www.crtech.com.

10.4.7 Balance Bar Plots

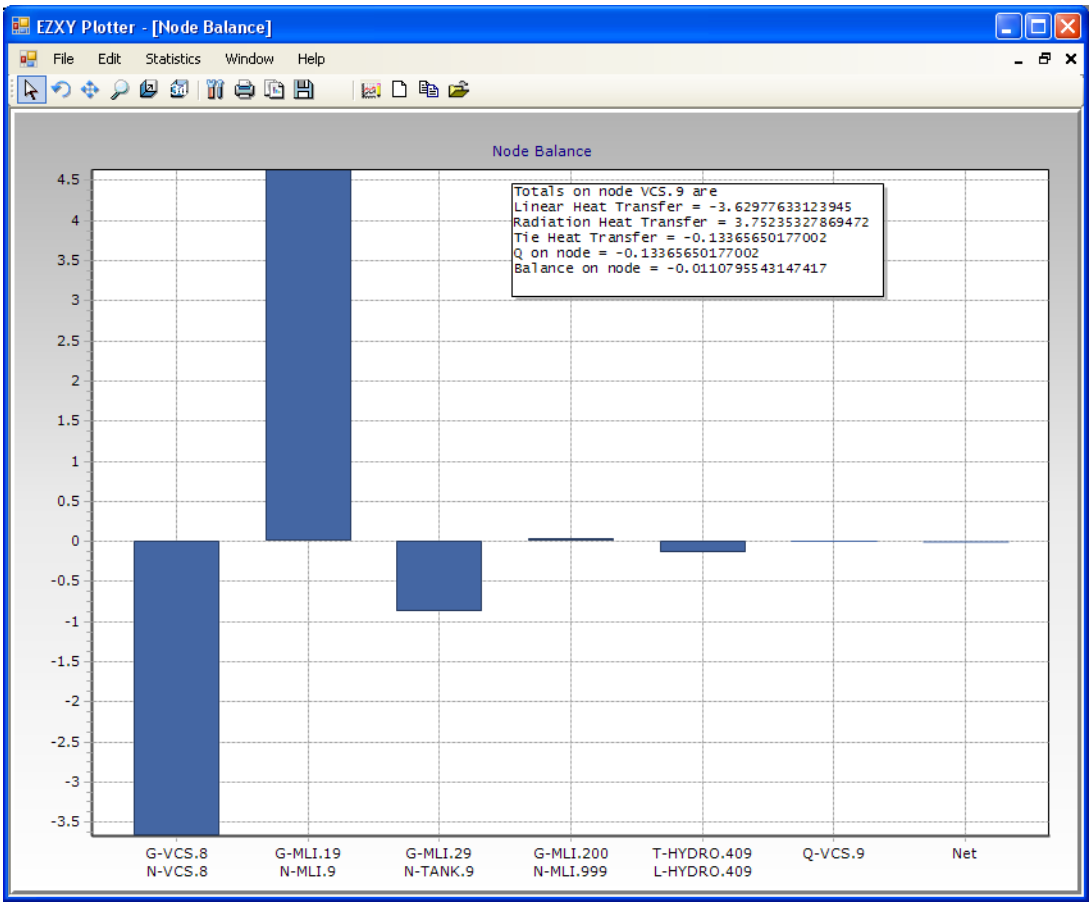
Bar plots are produced from node or lump balance commands. For example, a node energy balance plot is created by choosing the Post-processing>Node Balance option from the Network Diagram window. Bar plots are made from a single CSR folder or SAVE file at a single point within that file: the current time/record number chosen from the Case Manager, as described in Section 10.4.7. By selecting one or more nodes, then choosing Post-processing>Node Balance, an unique EZXY bar plot will appear for each node showing the details of energy flows in and out of each node, along with the net imbalance. Enough data must be saved to make such maps possible. For example, if temperatures are saved but conductances or heat rates are not, then the bar plot will not be possible. .

> Caution: selecting too many (or all) objects will result in an excessive number of bar plots that may exceed the memory or graphics limitations of some computers.

> Warning: Node and lump balance plots are generated from Sinaps based on data from the *network and post-processing data source*. Since the data does not exist directly on the data source, when a balance plot is saved the regenerate option will NOT update the plot to reflect changes in the save file. Consequently the plot must be recreated from Sinaps to reflect any changes between runs.

Figure 10-15 shows a sample node balance plot: a bar plot depicting energy flows into and out of node 9 in submodel *vcs*: node "vcs.9." The node is gaining heat through MLI conductor #19, but is losing even more heat through MLI.29 and other ties and conductors, for a net loss (as seen in the legend) of 8.4 Watts: the node is cooling down. (If it had been an arithmetic node, the net power is normally nearly zero).

FIGURE 10-15. Sample Bar Plot: Node Balance



For lumps, additional suboptions exist for bar plots:

Mass Flowrate. Mass flow rates in and out of the lump are tallied, with the net value representing the dM/dt: the change in mass for the lump at the current time.

Path Energy Rate. Energy rates entering and leaving the tank via paths are tallied. The net value shows the net imbalance due to advection, which might be countered by convection, storage, etc.

Tie Conductance. Tie conductances (UA) are summarized, showing the relative strength of convection if more than one tie exists on this lump.

Tie Heat Rate. Tie heat rates (QTIE) are summarized, such that the net value is the imbalance due to convection.

Energy Balance. This plot shows the net gain or loss to the lump and the routes (ties and paths) responsible for this imbalance. The net path advection imbalance is listed as a single value, “Net Path ER,” which is the same as the net value in the Path Energy Rate plot (described above).

CHAPTER 11

Export and Import

11.1 Exporting/Importing Models

11.1.1 Sinaps® vs. SinapsPlus®

Sinaps is a replacement for the original nongeometric SINDA/FLUINT graphical user interface, Sinaps*Plus*. The functionality of Sinaps*Plus* has been retained in all possible areas, although the program has been updated significantly providing a more user-friendly interface. Since Sinaps is a native Microsoft Windows™ program, the look and feel of the tool is more consistent with other Windows-based software. Sinas*Plus* was a cross-platform tool supporting both PC and Unix systems. (Unfortunately since Sinaps is Windows based, it is not available for Unix platforms.)

A major difference between Sinaps and the previous Sinaps*Plus* is the lack of an image file and associated binary archive files (*.im and *.bin respectively). Instead of using the binary archive to transfer files between machines and co-workers, the Sinaps *.smdl file (along with the thermophysical property file or include files, if any), can be directly transferred between machines. Also, double clicking on an *.smdl file opens the Sinaps model into a new Sinaps session.

Other major differences include the fact that Sinaps*Plus* had no parallel to the Case Manager nor to the Model Browser.

Also, in Sinaps, all of the network is in one diagram instead of the multiple linked diagrams used in Sinaps*Plus*. Because of this last difference, it is impossible to arrive at the same network diagram after import if there is more than one submodel or if a duct macro is used.

11.1.1.1 Exporting and Importing from SinapsPlus

To transfer files from Sinaps*Plus* to Sinaps, the export feature in the last version* of Sinaps*Plus* is required.

To export a model from Sinaps*Plus*, select the Model Control tab and click Utilities>export to Sinaps as shown in Figure 11-1. This will create an XML based *.smdl file which can be opened by Sinaps.

* Version 4.8, including the lastest patches if applicable. To export from earlier versions of Sinaps*Plus* requires first upgrading the model to this last version, since Sinaps export features were only available in the last release in order to provide an upgrade path

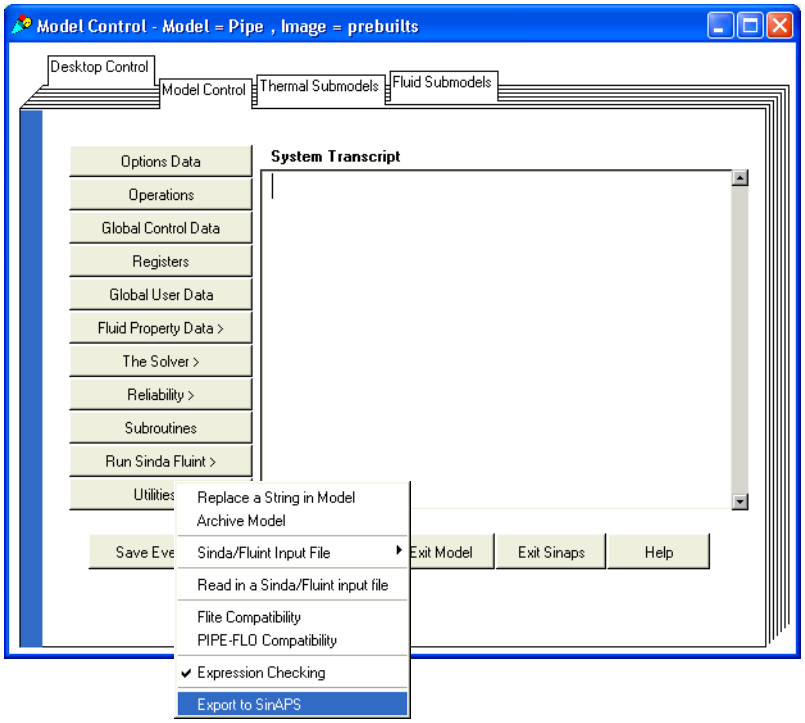
Fluid Property Descriptions: When exporting a model from Sinaps*Plus*, all fluid property (FPROP DATA) files that were added as INCLUDE files will be transferred. A fluid description defined by the user directly in Sinaps*Plus* will be exported as an INCLUDE file and associated INSERT statements automatically added to the model. The INCLUDE file created will be automatically named f[fluid ID].inc (e.g., "f8000.inc").

Network Diagram: The layout of the network diagram will be retained to some extent, except *it will likely appear different after importing* since all submodels now reside in a single network window (compared to the use of separate network windows for each submodel and for each macro in Sinaps*Plus*). During the export and import of a model, layers will be created for each submodel and each macro as a means to combine multiple networks into a single network. The user can turn these layers off/on in Sinaps to aid in adjusting the new network layout for the imported model.

LINE and HX Macros: LINE and HX macros will import to Sinaps as "Pipes." When importing from Sinaps*Plus*, any previous customization of the network elements within the macro will be retained. Note that after creating a macro in Sinaps*Plus*, all macro level information (such as length, increments etc. are lost. To prevent loss of data during export/import, the "Customized" check boxes in the pipe form will be activated for *all* imported macros. Unchecking this button and saving the form will result in lost data. Once imported, it is recommended that this button be left checked at all times.

Caution: Once a macro is created in Sinaps*Plus*, the original macro information was lost, such as register usage for length and diameter. Therefore when imported into Sinaps, register information for all parameters on the Data tab of the Macro Edit form will be lost, but can be redefined by the user after import. Note that register information on the individual network elements is retained, although it can be lost if the "Macro is Customized" check box is unselected by the user and the form saved.

FIGURE 11-1. Exporting from Sinaps*Plus*



11.1.2 Importing SINDA/FLUINT Text Files

Text-based SINDA/FLUINT files can be imported directly into Sinaps using the New>Read in a Sinda/Fluint input file option. This option should be used with care, and *it is **not** intended to be used to read models that were generated by Sinaps, SinapsPlus, or Thermal Desktop*. It is instead intended to be a one-time operation that moves a hand-built model into the Sinaps system, with enough time allocated for organizing the network depiction (which is otherwise absent in a SINDA/FLUINT input file). The user is strongly advised to save and close any open Sinaps models before starting a new read-in operation.

When an input file is read in, Sinaps will place the resulting files in the same location as the input file. It is recommended to copy the text input file and any supporting include files to the final destination folder before importing. Then start Sinaps, select Tools>Read in SINDA/FLUINT input file. When the browse window appears select the same final destination folder where the text input file was place.

Once an input file is selected, the user will be prompted whether or not the file should be saved first. Sinaps will automatically create a network diagram by laying out objects on a grid as it reads them. During readin, each submodel will automatically be placed on a separate layer to help the user in customizing the layout after model readin.

When the model is read in, the network will be created by laying out the objects on a grid as they are read in. When the readin is done the Sinaps will automatically launch the automatic layout feature which uses the Layered Digraph autolayout option. At the prompt "Layout can be time consuming - do you want to save first," the user can continue without saving (select no) or cancel the automatic layout option (select cancel). The user can toggle the layout to a Force Directed layout using the network menu option Layout>Automatic Layout>Force Directed. The force directed layout treats links on the diagram as springs and locates network objects so that all the links are of more or less equal length and there are as few crossing edges as possible. The layered digraph creates layers or tiers of network objects.

Importing Large Models: For large models, the user should consider isolating portions of the model that *shouldn't* be depicted using INSERT commands (see "INCLUDE/INSERT File Usage and Restrictions" on page 11-4) or the CSTARTIN and CENDINC commands (see "Using CSTARTINC and CENDINC" on page 11-4). Because the operation of INSERT commands can vary depending on where they are encountered in the input file, the user should also consider removing them *before* read-in, and replacing them in a controlled fashion *after* the model has been read in.

LINE and HX Macros: Line and HX macros will import to Sinaps as "Pipes." Sinaps currently does not allow the user to specify temperature, pressure or quality increments along a pipe when initializing since the feature was rarely used in the past. Therefore when imported into Sinaps, there is no means to pass this information (if used) along to the Sinaps pipe edit form. Additionally pipe level register/expression information is lost but is retained at the network element level. To prevent loss of data, when a macro is imported, the Customized buttons in the pipe form will be checked.

> Caution: When imported into Sinaps, the "Customized" buttons will be check to prevent loss of incremental temperature, pressure, and quality information along with register and expression information. Unchecking the box and saving the macro form will result in lost data.

Registers in GEN and MACRO statements: Sinaps does not support the use of registers in GEN statements for the number segments for LINE and HX macros or for the number to be generated in GEN statements. Prior to reading the text file into Sinaps, the user must go through the input deck and replace such usage.

Fluid Property Inserts/Includes: When Sinaps reads in a text based model it creates a combined.inp file (similar to what the preprocessor creates) and then reads the file in line by line. Since the “big” fluid property files are very large, this can result in the read in process taking a very long time. (The reference to big refers to property files which are available from CRTech which include the word big in their name.) If a model uses “big” fluid property descriptions, it is recommended the insert statements be commented out prior to the read-in process. After reading the model in, edit the model and use the Advanced tab of the Case Manager to access the Fluid Property Includes dialog and point to the appropriate fluid property files.

11.1.2.1 INCLUDE/INSERT File Usage and Restrictions

INSERT statements should be used in preference to INCLUDE statements. INSERT statements are expanded during the reading process and the data in INSERTed files will be a part of the Sinaps model. However, it is not a good idea to INSERT a large set of radiation conductors into Sinaps. Inserts of radiation conductors should be changed into INCLUDE's.

Do not use INCLUDE statements that contain partial node or flow data. Since INCLUDE statements are stored verbatim (i.e. the include operation itself is not performed) the information within the INCLUDE files will not be read in. A failure will occur if any conductors/paths/ties that are not defined in an INCLUDE file reference nodes/lumps that are defined in INCLUDE files.

INCLUDE statements may only appear in logic blocks, array data, user data, FPROP data, and conductor data. The read-in program will try to translate all data it finds in non-logic blocks (e.g., control constants in control data blocks). If it finds data that does not belong (like an INCLUDE) it will produce errors. The only parts of the input that are accepted unchanged are the logic and array blocks. Since Sinaps has a placeholder for conductor INCLUDE statements, these are also accepted.

In order for internal preprocessing operations to function correctly, INCLUDE files can only occur within recognized locations within node, conductor, flow data (lump), and FPROP data Sinaps options. If INCLUDE statements are made in these locations, and do not introduce new HEADER information inside of the INCLUDEd files, then preprocessing features will function correctly.

INCLUDE statements must reference whole files. Sinaps does not support inclusion of partial files (i.e., line number specifications).

When using the automatic INCLUDE file generation options (See Using CSTARTINC and CENDINC), FLUINT ties cannot connect to nodes that are not diagrammed.

11.1.2.2 Using CSTARTINC and CENDINC

One way to automate the process of splitting up a model into essential (diagrammed) parts and nonessential (non-diagrammed, INCLUDEd) parts is to use the CSTARTINC and CENDINC macro commands.

These commands are placed in the HEADER NODE DATA block(s) of the SINDA/FLUINT input file prior to reading that file into Sinaps. The nodes that are input following the command CSTARTINC and preceding its companion command CENDINC (both starting in column 1) will be placed in an include file called smnNode.inc, where “smn” is the name of the current thermal submodel. All other nodes will be read into Sinaps and will appear on a diagram.

All conductors connected to nodes that are not diagrammed (i.e., that appear between a CSTARTINC and CENDINC pair) will be detected and automatically placed in an include file named smnCond.inc, where again “smn” denotes the thermal submodel name of the current CONDUCTOR DATA block.

There can be multiple pairs of CSTARTINC and CENDINC within a file and within a single NODE DATA block. If CSTARTINC is active when the end of a NODE DATA block is encountered without a CENDINC being input explicitly, the terminating CENDINC is understood. However, it is recommended that it be input explicitly for clarity.

The generated INCLUDE file names will appear automatically in the INCLUDE list for the respective diagram. These files will be located in the same subdirectory as the image file. Changes to the locations of these files must be accompanied by a change to the INCLUDE list in each diagram, otherwise Sinaps will be unable to locate the file.

An example of using CSTARTINC and CENDINC is listed below:

HEADER NODE DATA, PAYLOD1

```
C
CSTARTINC
      100, 70., 0.200 * 0.21 $ RADOME
      101, 70., 1.071 * 0.21 $ COVER
CENDINC
C
      102, 70., 2.980 * 0.21 $ FEED #1
      103, 70., -1.0 $ ERN
C
CSTARTINC
      105, 70., 1.300 * 0.21 $ COVER
      106, 70., 1.700 * 0.21 $ COVER #2
CENDINC
C
      108, 70., -1.0 $ FWD COVER
      109, 70., 2.500 * 0.21 $ FEED #2
      110, 70., -1.0 $ ERN
```

At least two INCLUDE files would be created in the above example. The first, PAYLOD1Node.inc, would contain the non-diagrammed nodes 100, 101, 105, and 106 in ASCII format. The second, PAYLOD1Cond.inc, would contain all the remaining conductors in submodel PAYLOD1 that connected to these non-diagrammed nodes. In fact, if other submodels contained inter-model conductors that attached to the nodes PAYLOD1.100, PAYLOD1.101, PAYLOD1.105, or PAYLOD1.106, additional smnCond.inc files would be created to contain those conductors as well.

Caution: Unlike inter-model conductors, FLUINT ties cannot attach to INCLUDEd nodes. If CSTARTINC and CENDINC bracket a node that is tied, an error will occur during read-in and the read-in will be aborted.

11.2 Submodel Import and Export

Submodels networks can be imported and exported in Sinaps allowing a submodel to be transferred between multiple models. Submodel export and import is performed using the Model Browser. To aid in exporting, a list of export limitations is provided below.

1. Only one submodel can be exported at a time.
2. This export/import feature is restricted to the network.
3. Submodel data in the case manager is not exported. Note this include any logic block in the case manager.
4. Network element logic is exported with the network.
5. Expressions in data fields are retained, however registers referenced in the expressions are not automatically exported. Registers can be exported through the register dialog.
6. Tables references are exported but the tables are not. Tables can be exported through the table manager.
7. Any links (conductors, paths, etc.) across submodels will be dropped. Fluid submodels with *walled pipes* (macros) will have the ties dropped and be converted to *pipes without a wall* since the thermal side is a separate submodel.
8. Network objects such as comments, lines, and collections are owned by the network (not by any submodel) therefore not exported. Note that submodel elements in a collection will be exported.
9. Working fluid associations will not be retained for fluid submodels. Upon importing a submodel, Sinaps will prompt the user to associate a work fluid for the new submodel.

To export a submodel, highlight the submodel name in the submodel tree of the model browser. Using the main menu click the `Model>Export` option. A file will be created in the local working directory by default with the name *[submodel name].xml.subm*. The .xml.subm suffix is recognize by Sinaps as an exported submodel file. To import a model select the `Import` option and browse to the submodel file to be imported. Note that to import a submodel you cannot have an existing submodel with the same name. If the intent is to add the imported model to second submodel, after the import process is complete, the imported network can be “moved” to a different submodel.

11.3 Table Import and Export

For details on importing and exporting Sinaps Tables, please see “Importing and Exporting Tables” on page 5-128.

11.4 Register Import and Export

For deatils on importing and exporting registers, please see “Importing/Exporting Registers” on page 6-5.

CHAPTER 12 Tutorials

Several tutorials are provided in this chapter to assist the user in learning to use Sinaps. Some of the tutorials will cover how to work within Sinaps while other tutorials provide sample problems with model development.

In addition to these tutorials, there are several sample models available which correspond to the SINDA/FLUINT Sample Problems Appendix. These models can be found in the Sinaps install directory in the “Samples” folder. Please see the SINDA/FLUINT Sample Problem Appendix for a detailed discussion of these models.

The following tutorials are available. Each tutorial is designed to demonstrate various modeling methods and capabilities within Sinaps. When working through the tutorials, it may be preferable to open the PDF version of this manual accessible through the start menu (Start>All Programs>Sinaps>Sinaps Help PDF File.

Creating a Model Template

- Template creation
- Setting units and model defaults
- Creating and using registers
- Setting preferences for hover text

Customizing the Work Space

- Window management
- Docking of task panels
- Auto-hide of docked panels
- Custom toolbar creation

Cryogenic Tank Support - Basic SINDA Model

- Creation of nodes and conductors
- Use of thermophysical properties database
- Creating multiple submodels
- Creating radiation conductors
- Setting network object defaults in diagram

- Basic use of Case Manager for steady state simulation
- Basic color post-processing

Cryogenic Tank Support - Transient SINDA Model

- Applying a heat source to a node
- Use of conditional statments, expressions, and processor variables
- Basic use of Case Manager for transient simulations
- Renaming and copying cases in the case manager
- Transient plotting
- Customizing plot and axes titles

Cryogenic Tank Support - Temperature-dependent Conductivity

- Pointing to a new thermophysical property file
- Use of temperature-dependent properties
- Plotting from multiple sources
- Customizing plot labels and legends

Cryogenic Tank Support - Parametric Sweep

- Adding a new case
- Using parametric options in the Case Manager
- Plotting with a registers on the x-axis
- Text post-processing

Flow in a Pipe - Basic FLUINT Model

- Creating and using register groups
- Creating a Pipes
- Adding insulation to pipes
- Creating multi-conductor objects
- Using post-processing meters and gauges
- Adding elevation and gravity to a fluid model

Filling Scuba Tanks - FLUINT Model

- Using background graphics
- Renaming submodels
- Using network zoom
- Importing registers
- Renaming cases
- Using case specific register overrides
- Plotting for lump mass

- Launching multiple cases
- Plotting from multiple save files
- Customizing plot axes label formats
- Customizing labels in the EZXY Plotter

12.1 Creating a Model Template

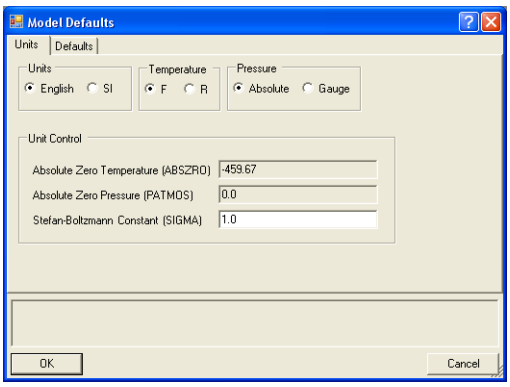
This tutorial will walk the new user through the initial steps to create a model, setting units and model defaults, and defining preferences. The final template file can be used for starting other tutorials and for starting other models. It may be useful to create two template files, one using ENG units and one for SI units.

Creating a Model Template

1. Open Sinaps by double clicking on the desktop icon or using the start menu **Start>All Programs>Sinaps**.
2. Start a new model by selecting **File>New>SINDA/FLUINT Model** or by clicking on the New icon. You will be prompted to select a location for the model files. Browse to the desired location, possibly creating a new directory, and then click **OK.** Click **OK** if you see the Information window below:

Important: The default file naming scheme can result in the same file names for different models. It is recommended that each model be created in a unique directory. If the model is opened in an existing subdirectory already containing another Sinaps model, the above message may not appear since the new model will automatically reuse the existing Td-Thermo.tdp file.

3. The *Model Defaults* window appears with the *Unit* dialog displayed.



4. For Units, select **English**
5. For Temperature, select **F**
6. For Pressure, select **Absolute**

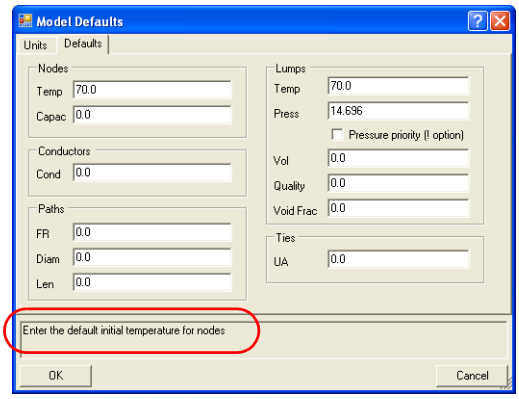
You may find it convenient to create two templates, one with SI units and one with ENG units. For this tutorial we will create a template using ENG units and then modify it to create an SI template.

The absolute zero temperature and pressure fields under Unit Control will automatically be set based on the radio button selection.

The Stefan-Boltzmann constant is defaulted to a value of 1.0. This assumes that any imported or user created radiation conductors will already have the constant factored into them.

Creating a Model Template

7. Click on the *Defaults* tab to make the dialog active.



8. Place the cursor over the Node Temperature data field and click the middle mouse button.[a] In the subsequent menu select Edit Registers. The Registers dialog (show at right) will appear.
9. Create the following registers of type "floating":

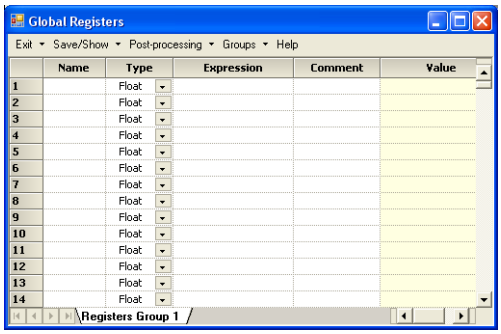
Name	Expression	Comment
Tinit	70	Initial temperature
Pinit	14.696	Initial pressure
ALinit	0.0	Initial fluid void fraction

Click **Exit>Save and Exit**

Defaults can be set for the various network elements to aid the user in setting initial conditions and setting key parameters.

Notice that by hovering the mouse over each input field, a description for the data field will appear in the bottom portion of the window

It is often convenient to set the defaults using registers so changes can easily be propagated throughout a model. We will create a few registers for common input parameters and then apply these registers as our default values.

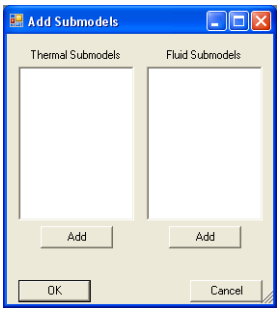
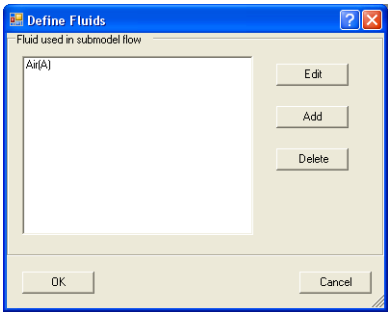


Since registers default to type "Real", it is not necessary to input a decimal point. Sinaps will convert the inputs to real when writing out the input file.

10. Highlight the contents of the Node Temperature field. With the contents highlighted, click the middle mouse button and select **Registers>Tinit**
11. Repeat this process assigning the lump temperature as Tinit, the lump pressure as Pinit and the quality as ALinit.
12. Click **OK** to save and exit the dialog.

Now apply the created registers to the model defaults.

Since we are creating a starting template we will not assign flow rate values or any geometry based values.

Creating a Model Template	
13. The Add Submodels window will appear  14. On the left side of the forum, click **Add** to add one thermal submodel. When prompted for the name type in **THERMAL**. 15. On the right side of the forum, click **Add** to add one fluid submodel. When prompted for the name type in **FLOW**. 16. Click **OK** to exit form.	Create one thermal and one fluid submodel for the template. When using the template for subsequent models, additional submodels can be added. Empty submodels should be deleted if not used. Submodel names may be any *unique* character alphanumeric string of up to 32 characters and must start with at least one alphabetic character (A through Z). Note unique means no two submodels may share the same name, even if one is a thermal submodel and the other is a fluid submodel.
17. The Define Fluids dialog will appear.  18. Click **OK** to dismiss (accepting air as the working fluid).	Define air as the working fluid for the fluid submodel. The working fluid can be changed later through the model browser.
19. The network window should automatically open. If it does not, open it by clicking **View>Network** 20. Open the Model Browser if it is not already visible, **View>Model Browser.** 21. In the Model Browser double click on **Registers** to open the Registers window. You should see the registers we previously created when we set up our model defaults.	Alternatively, the network can be opened by double clicking on “Network” in the Model Browser window. Alternatively, the registers can be opened through the View>Registers menu.

Creating a Model Template	
22. Open the Preferences dialog by clicking **Tools>Preferences** 23. In the Preferences dialog, click on the **Hover Text** tab. 24. Use the pull down menu options to select the desired parameters to be displayed in hover text for each type of network element. 25. Uncheck the **Show Post-process Information** box if you only want to see input parameters as hover text at all times. If the box is left checked, the hover text will display values based on color post-processing selections in the Case Manager. 26. Click **OK** to dismiss the window.	Hover text will appear in the network diagram when the cursor is placed over any network element (object or link). The preferred hover text content can be set by the user through the preferences dialog (menu Tools>Preferences). For detailed information on the preferences dialog, please see "Setting Preferences" on page 3-11.
27. Click the menu **File>Save As** to save the template and provide the name **templateENG.xml.smdl**.	
It is useful to have two template models, one in English units and one in SI units. Next we will modify the template to create a version using SI units.	
28. Click the menu **File>Save As** and provide a new name for the second template, perhaps templateSI.xml.smdl.	Save under a new name.
29. Click on the menu option **Model>Defaults and Units**. 30. On the **Unit** tab, select the **SI** option for Units (the value of ABSZRO will automatically change to -273.15 representing degrees celsius) 31. Move to the **Defaults** tab. Note that all defaults have been reset since the model units were changed and must be redefined. 32. Set the node and lump temperature defaults to **Tinit** 33. Set the lump pressure to **Pinit** 34. Set the void fraction to **ALinit** 35. Select **OK**	Use the advanced case set dialog to access Global control data and reset units for the model to SI and degrees C, retaining absolute units. *Important: Due to the complexity of tracking units in expressions and user defined logic,* ***Sinaps will not automatically convert inputs when the units are changed.*** *It is up to the user to modify the model appropriately for any unit changes.* *Important: In general once a model has been created, changing units is* ***not*** *recommended.*

Creating a Model Template	
36. Open the Model Browser, **View>Model Browser** (or click Model Browser tab in the task panel if already open) 37. Double click on **Registers** 38. Change Pinit to **101325** 39. Change Tinit to **20** 40. Select the register menu item **Save/ Show>Save & Show Values**	Propagate the unit changes through the registers. The pressure will be changed to 101325 Pascals representing atmospheric pressure and 20 degrees Celsius. The "Save & Show" option will resolve all expressions and show the values for each register. Note, Sinaps may be unable to resolve some expressions, perhaps based on processor variables, in which case a random number value may be assigned.
41. Click the menu **File>Save** to save the template.	

a. Like all tutorials in this document, this tutorial assumes that the user has access to a 3-button mouse, or that they can equivalently program the middle button.

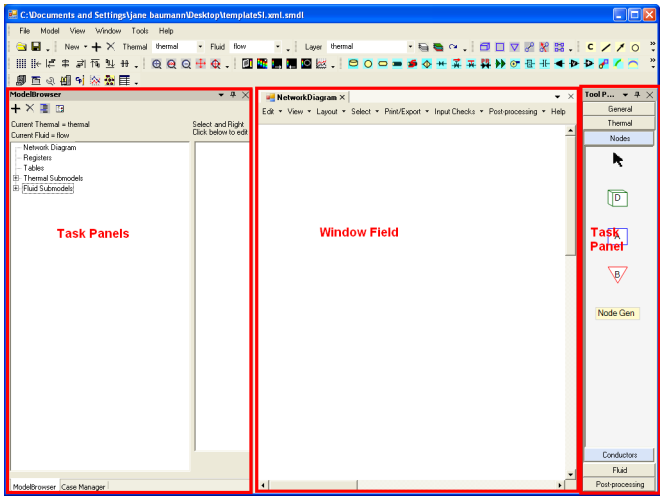
12.2 Customizing the Work Space

This tutorial will introduce the user to window management, docking of task panels and customizing toolbars.

Custom Workspace	
1. Copy one of the template files created in the last tutorial to a new folder (directory).	The default file naming scheme can result in the same output file names for different models. It is recommended that each model be created in a unique directory.
2. Open the model by double clicking on the file name.	A message will appear stating "Properties not found in specified location or locally: TdThermo.tdp" warning you that the thermophyspical properties database could not be found. Click OK to continue. A second message will appear stating "ThermoPropDB: filename TDThermo.tdp does not exist, it will be opened in the current working directory" letting you know a new empty thermophysical property database will be created. Click OK to continue.

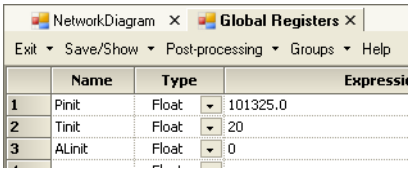
Custom Workspace

The default layout should appear similar to the figure below with the Case Manager and Model Browser *task panels* docked on the left side of the frame, the tool pallet *task panel* docked on the right side of the frame, and the network diagram visible in the *window field*.



Window layout options in the window field will be demonstrated first...

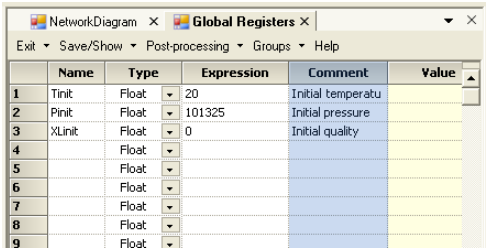
3. The **Network Diagram** window should be visible. If not, click the Network icon ( ) to open the network window.
4. Click on the register icon ( ) to open the **Global Register** dialog.



Open the register dialog to so multiple windows are active in the window field.

After opening the registers, the global register window and the networkdiagram window should be appear as tabbed windows in the window field

How windows are displayed in the window field can be customized using the Preferences dialog. The two options are tabbed or floating. Next we will explore the floating and tabbed windows.

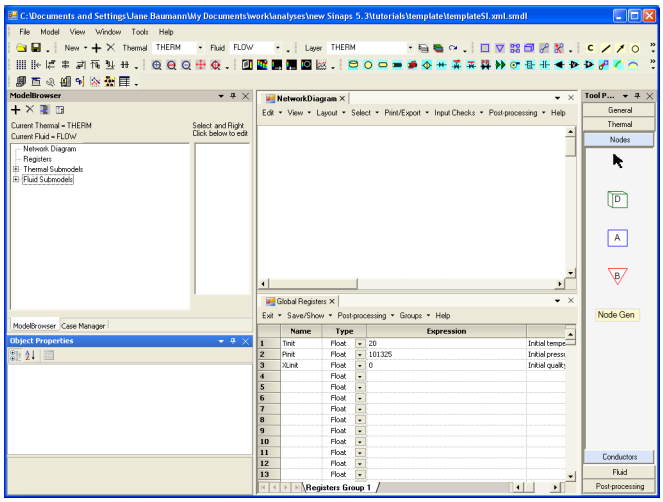
Custom Workspace	
5. Open the Preferences dialog by clicking **Tools>Preferences** 6. Under Window Management select **Floating** 7. Click on the top frame of the network or register window. Holding the cursor down, the window can be moved in the window field. 8. Clicking on the buttons in the upper right of the network or register windows ( ) will minimize, maximize or close the window. Click on the **X** to close the network window. 9. Reopen the network window using the menu **View>Network** or click the Network icon ( )	By default the window management is set to tabbed. Alternatively floating windows can be used.
10. Click the menu **Window>Cascade** 11. Click the menu **Window>Tile Vertical**	To aid the user in managing floating windows, the windows can be cascaded or tiled. When four or more windows are open, tiling will result in tiling both vertically and horizontally.
12. Open the Preferences dialog by clicking **Tools>Preferences** 13. Under Window Management select **Tabbed** *Important: When several windows are open such as the network, registers, arrays, and logic blocks, tabbed window management provides an efficient means to move between windows.* 14. Click **OK** to dismiss the preferences dialog. 15. Move back to the register window and hover the cursor over the tab of the register window and click the right mouse button. A menu will appear. Select **New Horizontal Tab Group** *Important: Using horizontal/vertical tab groups allows multiple windows to be viewed at one time. When a window is opened, it will open in the tabbed group which is currently active.*	Explore working with tabbed windows. Tabbed windows will appear maximized in the window field with labeled tabs across the top. The user can easily move between windows by clicking on the tabs. 

Custom Workspace

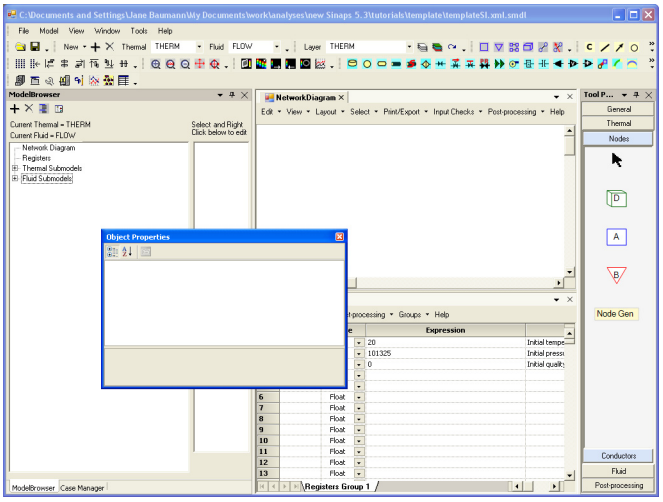
Docking and undocking of task panels will be covered next...

Any of the task panels can be docked or undocked as discussed in this next section. When undocked, the panels appear similar to other floating windows such as the network and registers, however they are not confined to the window field of the work environment. The task panels include the Model Browser, Case Manager, Object Properties, Tool Pallet, and Thermophysical Properties.

16. Use the menu **View>Properties** to open the Object Properties dialog. The properties should display below the Model Browser and Case Manager docked to the left frame as shown below.

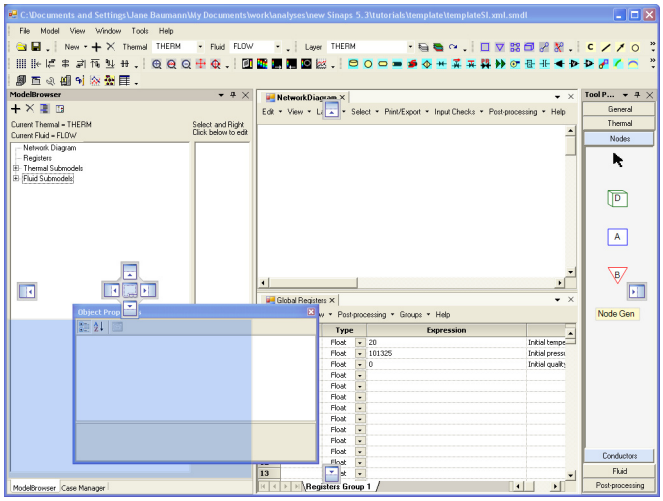


17. To undock the properties task panel, click on the top blue frame portion of the window and holding the left mouse button down, drag the window away from the frame. Release the mouse button. The window should appear as shown below. The window is now undocked and can be placed anywhere.



Custom Workspace

18. To redock the window, grab the top frame and (holding the left mouse button) down drag the window toward the left frame. Docking icons will appear on the diagram. When the cursor is hovered over various docking icons, the pending location of the docked panel is highlighted in blue. In the graphic below, the original location is highlighted.

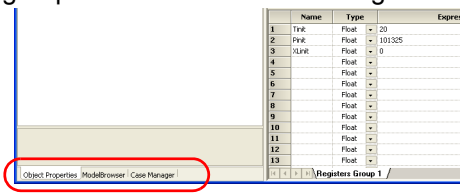


Normally five docking icons will display: one for each side of the Sinaps window frame and the fifth icon for multipanel or *group* docking. As an undocked panel is moved around the work space, the group docking icon will appear over the various work areas.

19. Place the cursor over each of the various docking icons to see the highlighted locations. Notice that the panel can be docked to the top, bottom, left or right of the main Sinaps window. It can also be docked as a tabbed group with the Model Browser and Case Manager, or the Tool Pallet task panel. When docked as a group, the panels can be docked individually to the group frame or tabbed with the group.

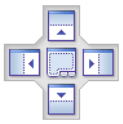
20. Dock the Object Properties back in its original location.

21. Undock the Object Properties panel again and redock it as a tabbed group with the Model Browser and Case Manager by locating the cursor in the center of the group dock icon shown at the right.

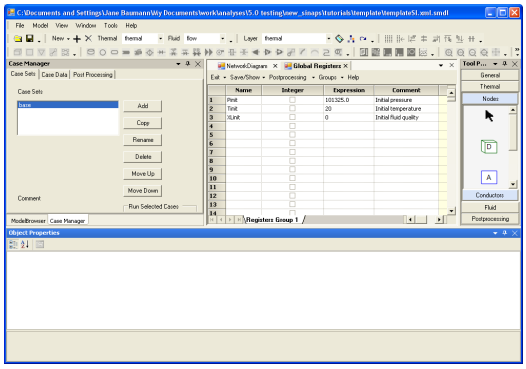


22. Undock the properties panel from the group by grabbing the tab labeled “Object Properties” and dragging it out of the group.

The group docking icon is shown below:



When docked as a group, the entire group can be undocked or docked by grabbing the top frame of the group. To undock a single window from the group, grab the tab.

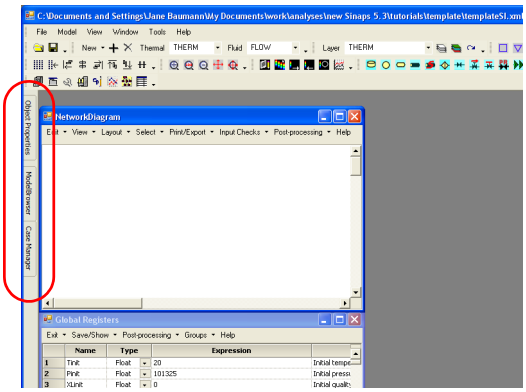
Custom Workspace	
23. Dock the properties panel to the bottom frame of the Sinaps window. It should appear as shown at the right.	
24. Undock the properties dialog again and this time dock it as a window tabbed with the network diagram and the register window using the center “tabbed group” icon. 25. Once docked in the window field, the panel becomes a tabbed window and is no longer dockable. To make it dockable again, right click on the title bar of the object properties window and select **Dockable** 26. Dock the Object Properties form back in its original location (lower left).	Task panels can also be docked to the frame of the *window field* or tabbed with the other windows in the window field. When tabbed in the window field it will behave in accordance with the window management selected in the preferences dialog (step 6 above).
27. Use the menu option **Tools>Preferences** to open the preferences dialog. 28. Set window management to either tabbed or floating. 29. Set Panel Docking State to “Save” to save docking states between Sinaps sessions.	Docking states can be saved using the preferences dialog.

Custom Workspace

The next section will cover auto-hiding of docked windows...

30. Click on the pin icon in the upper right corner of the object properties dialog. The panel will collapse to a tab on the left frame of the Sinaps window.
31. Click on the pin icon for the Model Browser/case Manager group. It will collapse to a group of tabs.
32. Hover the cursor over each of the three tabs to see how the panels will fly out for access.
33. Pin the Case Manager back into place. Notice that this pins the panel group, not just the case manager.

When docked, task panels display a pin icon in the upper right corner of the panel. This icon is used to collapse the panel into a tab on the frame of the Sinaps window. This conveniently maximizes the work space for the network window. Alternatively, click the right mouse button on the upper frame of the panel or group and select **Auto-hide**.



Once collapsed, the panels can fly out by hovering the cursor over the tab. Clicking the pin icon again will fix the panel back into its stationary position.

Custom Workspace	
This next section will cover customizing toolbars...	
34. Locate the right portion of the “View” toolbar. (Make sure the Network window is open.) 35. Right click on the down arrow at the right end of the toolbar and select **Customize** 36. Move to the **Commands** tab 37. In the Categories window located **Network Diagram**. If it is not expanded, expand it by clicking on the “+” icon and select **View**. Commands associated with the Network View menu will appear in the right pane. 38. In the Commands window, left click on the icon for **Link Label Toggle** and drag it into the View toolbar 39. Click **Close** to exit the customization dialog.	For this portion of the tutorial, a new icon will be add to the main “Fluid” toolbar. Custom toolbars define your preferred Sinaps interface, and will reappear the next time Sinaps is opened, including when opening an old model. Commands in the customization dialog are organized based on the menu systems. There are two menu systems: the Sinaps main window menus, and the network menus. The categories window on the left side of the Command tab select which menu system you are working with, the right window offers the associated commands. New toolbars can be created. However, adding objects to existing toolbars is often the best approach.

12.3 Cryogenic Tank Support - Basic SINDA Model

Consider a cylindrical rod with constant thermal properties (ρ = 8000 kg/m^3, k = 15 W/m-K, C_p = 500 J/kg-K) that is coated with a paint whose infrared emissivity is ε = 0.3. The length of the rod is 1.0 meter, and the diameter is 1 cm.

The rod is used to suspend a 40K cryogenic vessel inside of a 300K vacuum chamber. For the purposes of this problem, the effective radiation sink temperature within the vacuum is specified to be 110K. See Figure 12-1.

What is the heat leak into the vessel?

The problem can be solved entirely by steady-state solutions, since no transient event exists. The rod may therefore be represented by zero capacitance (massless) arithmetic nodes. (In steady state solutions, diffusion nodes are treated as arithmetic nodes, so their capacitances will therefore be ignored in this case.)

FIGURE 12-1. Cryogenic Tank with Support Strut

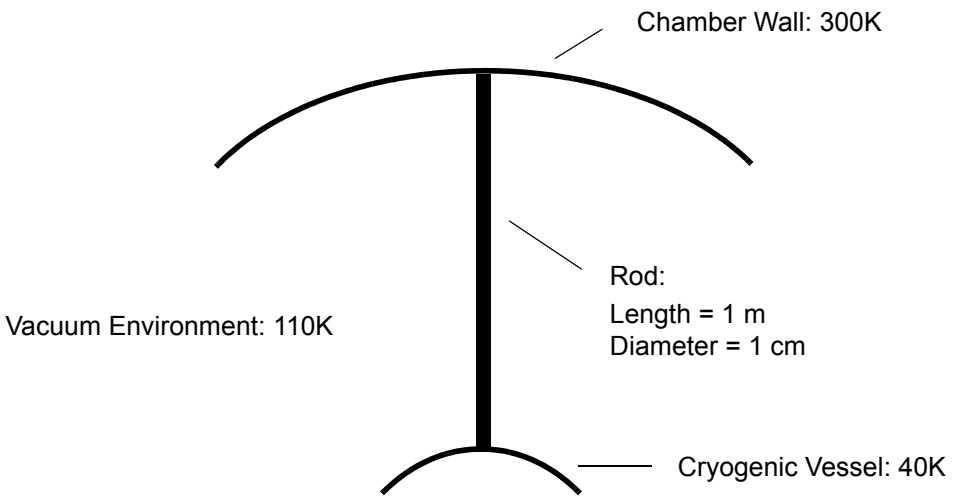
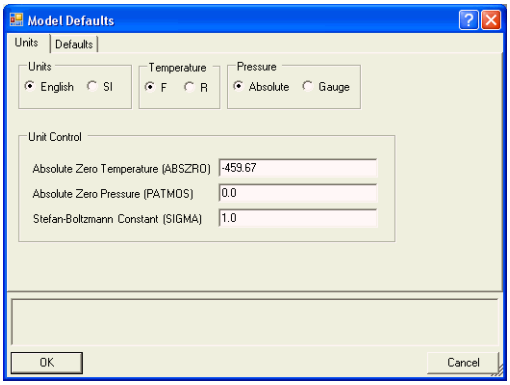
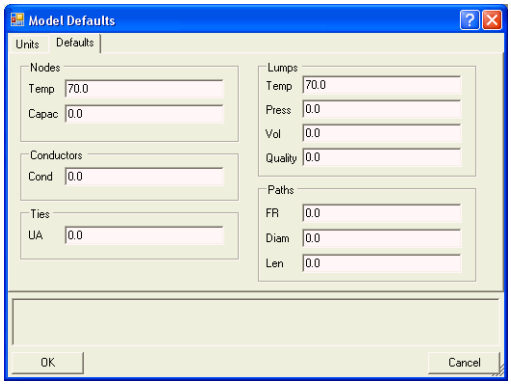


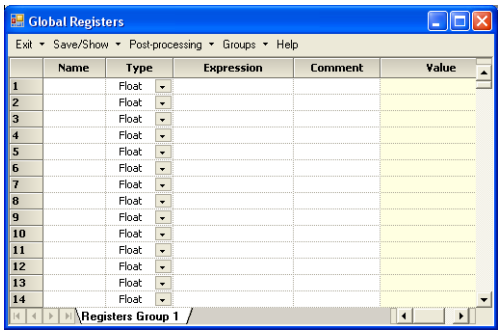
Figure 1: Sample Problem Schematic

Cryogenic Tank Support	
1. Open Sinaps by double clicking on the desktop icon or using the start menu **Start>All Programs>Sinaps.** *Important: The default file naming scheme can result in the same file names for different models. It is recommended that each model be created in a unique directory.*	
2. Click the menu item **File>New>Sinda/ Fluint Model** 3. Browse to a location for the model directory, any will do. Click **OK**.	Create a new model.
4. The *Model Defaults* window appears with the *Unit* dialog displayed. 	Our sample problem has been defined in SI units with temperatures specified in degrees Kelvin. We could have created the model in English units, but would then need to convert our inputs. So instead, we will create the model using the same units in which the problem has been defined.

Cryogenic Tank Support	
5. Under Units, select **SI** 6. Under Temperature, select **K** 7. Highlight the contents in the *Stefan-Boltzmann Constant* data field and then press the middle mouse button. On the pop-up menu, select **Built In>sbconsi**	The Stefan-Boltzmann constant is defaulted to a value of 1.0. This assumes that any imported or user-created radiation conductors will already have the constant factored into them. In this model we will be creating all the radiation conductors so we will provide a non-unit value for the constant that will be applied to all radiation conductors in the model. To define the value, we will use the built-in constant *sbconsi*.
8. Click on the *Defaults* tab to make the dialog active.  9. Place the cursor over the Node Temperature data field and click the middle mouse button. In the subsequent menu select **Edit Registers**.	Defaults can be set for the various network elements to aid the user in setting initial conditions and setting key parameters. We will use the defaults dialog to set values for nodes and conductors. We can initialize the support strut to either the tank, chamber or radiation environment temperature. For this tutorial we will use a value of 40K.
Important: Sinaps allows the user to create user-defined variables called "registers." These registers can be used in any data field or logic block and provide a means for easily modifying a model. Changing the value of a register automatically propagates the change throughout the model, both before or during the solution.	

Cryogenic Tank Support

10. The Registers window appears.



11. Create the following registers setting the type to “Float”:

Name	Expression
tinit	40
length	1.0
diameter	0.01
num	10

Comments can be added if desired. Registers are not case sensitive.

12. In the registers window select **Exit>Save and Exit**

To create a register, provide a name, an expression, and an optional comment. Expressions can be a numeric value, another register, or an expression-based on user-defined registers, built-in registers, and built-in functions. Registers can also be based on processor variables.

All registers are of type float (floating point) unless the type pull down is used to specify integer (Int) or double precision (Double). See “Defining Registers” on page 6-2 for detailed information.

The value column displays the value of the expression and will be updated by Sinaps upon saving the values.

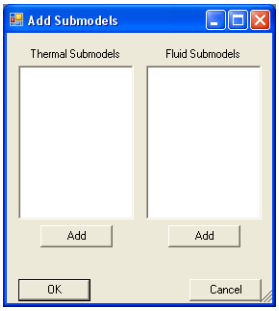
By setting the resolution along the pipe (NUM), the conductances etc. can be set up based on registers. However, any changes to the register representing the resolution along the strut will require changing the network accordingly. In other words, nodes and other network elements will not be automatically added or deleted based on a register value.

13. Upon exiting the Registers window, the defaults dialog will become active again.
14. In the group **Nodes**, set the **Temp** (temperature) to **Tinit**
15. Under Conductors, set the **Cond** (conductance) to **.25*pi*diameter^2/(length/num)**
16. Click **OK** to save and exit form

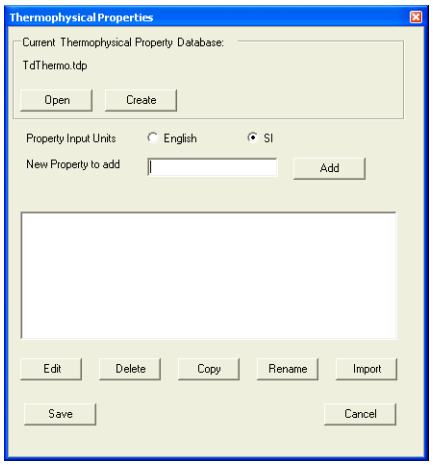
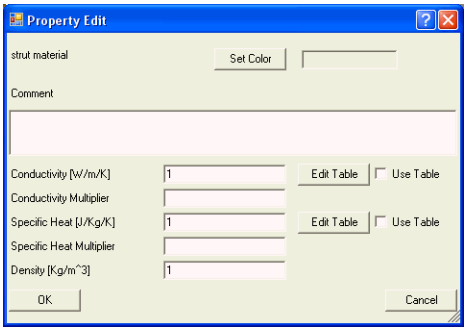
These defaults will remain in place for all network objects created until the user changes the defaults. The defaults menu can be accessed through the main menu **Model>Defaults**.

Note that the “conductance” being defined is not a true conductance (with units of power per degree). Rather is an aspect ratio only, with units of length. In other words, if conductance is calculated as k*A/L, then only “A/L” is being supplied. This is because the “k” will come from a material property to be defined separately.

Cryogenic Tank Support

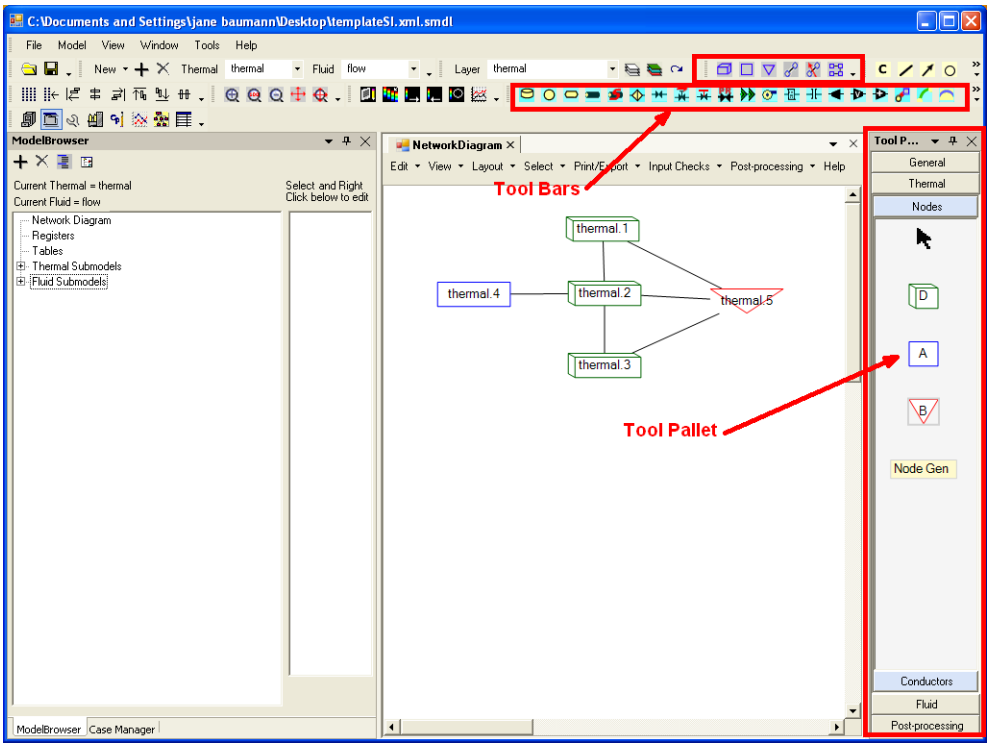
17. The Add Submodels window will appear.  18. Click the **Add** button under the Thermal Submodels window. In the "New sub-model name dialog, enter **strut** to create a thermal submodel. The new submodel will appear in the Thermal Submodels list. 19. Click **OK** to exit the form.	Define a thermal submodel for the strut network.

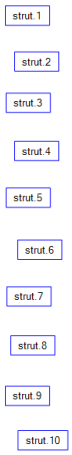
Important: It is always recommended that the user save often during model development to prevent loss of data. To save your model at this point use the main menu option ***File>Save****. You will be prompted for a location and file name. Type in the file name* ***strut.xml.smdl****. Click Save to exit the dialog.*

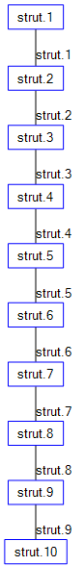
Cryogenic Tank Support	
20. Open the Thermophysical Property dialog using the main menu selection **View>Thermophysical Properties**  21. In the New Property to Add field input strut material then click **Add** 22. The Property Edit window will appear.  23. Under Conductivity enter **15** 24. Under Specific Heat enter **500** 25. Under Density enter **8000**	"Thermophysical properties" refers to density, specific heat, and thermal conductivity. Using the thermophysical properties database, users can create a shared property database to ensure consistent property definitions between models. The property file TdThermo.tdp is automatically generated by Sinaps. If the user has a existing property file, the Open button can be used to browse to the file, thus linking the file with the model. When defining units in the Thermophysical Properties dialog, the units do *not* need to be consistent with the model units, however the user must select either English or SI and all properties must be defined in the *same* units. This tutorial will use SI. Create a new material property named "strut material" and provide the properties as stated in the problem description (ρ = 8000 kg/m3, k = 15 W/m-K, C_p = 500 J/kg-K) In the Property Edit dialog, the comment field can be used to document items such as references for the properties being used. Temperature-dependent properties can also be defined, as will be demonstrated later. "Multiplier" fields are provided to assist in parameter studies to easily scale a property value. Using a register as a multiplying factor and then varying the register during a parametric run will result in a sweep of property values.
26. Click **OK** to close the Property Edit window 27. Click **Save** to save the Thermophysical Properties dialog. 28. Click the **[X]** in the upper right corner of the Properties dialog to dismiss the window.	The thermophysical property is a dockable window, similar to the case manager or the model browser. It can only be dismissed by clicking the [X] in the upper right corner. As an alternative it may be docked for future access.

Cryogenic Tank Support	
29. Click **Network Diagram** icon ( ) to open the network window	The network diagram is where the sketch of the model will be created.

When creating the network diagram, network objects can be created using either the tool bars or the tool pallet, highlighted below. Some users find the tool bars easier to use while others prefer the accordian panel tool pallet.



Cryogenic Tank Support	
30. Create nodes to represent the strut. Click on the arithmetic node icon in the tool bar (or use the equivalent option in the Tool Pallet in the Thermal/Nodes group). 31. Place the cursor in the network window where the first strut node is to be placed and click the left mouse button. A node will appear in the diagram named “strut.1” 32. Repeat clicking the cursor in the network window until a line of 10 arithmetic nodes have been created. *Important: Continuing to click the left mouse button in the network will result in additional nodes being dropped onto the network. To exit object creation and enter object select mode, click the right mouse button on any blank portion of the network.* 33. The diagram should appear similar to the diagram at the right (location dependent).	Because of the long aspect ratio of the rod, this is clearly a one-dimensional problem. The rod will be divided into 10 equal lengths, represented by nodes numbered 1 through 10. (For demonstration purposes, the discretization of the rod has been purposely underestimated.) Based on the default settings, these nodes will be created using a temperature of 40K. 
34. Conductors will now be generated. Move the cursor over the first node created (near edge of node icon) until the cursor changes to a hand ( ). **Press and hold** the left mouse button and **drag** a line to the second node. **Release** the mouse button. A line appears between the two nodes representing the conductor. *Important: If you find it difficult to access the link drawing mode, holding the shift key while drawing a link will force the mouse into link drawing mode.*	There are two types of heat flow paths: axial conduction along the rod, and radiation exchange between the surface of the rod and the vacuum environment. Linear conductors will be used to represent solid conduction along the rod. The conductance of the conductors will automatically be assigned based on the default value.

Cryogenic Tank Support	
35. Repeat the previous step between each pair of nodes along the length of the strut to create a series of conductors between all the nodes. 36. Link (conductor) labels can be turned on. Using the network menu, select **View>Link Lable Toggle.** 37. Select everything (drag a window). Using the network menu, select **Layout>Align Vertical Centers**. 38. Everything should still be selected. Using the network menu, select **Layout>Even Space Vertical**. 39. Click the **Escape** key to unselect everything. 40. The diagram should appear as shown to the right.	
41. From the main menu, select **File>Save**	Save the model.
42. If the Model Browser dialog is not open (left panel, possibly tabbed with the Case Manager) use the menu option **View>Model Browser** to open it.	The Model Browser provides a tree structure of the entire model. It also provides access to the network and register windows.
43. To create a new submodel use the main menu and select **Model>Add** or click the Add icon . 44. A prompt will appear asking if it should be a thermal or fluid submodel. Select **Thermal.** 45. The Submodel Name dialog will appear. Enter **environment** and click **OK.**	Any master model is comprised of one or more submodels. Submodels help organize network elements within a master model. The system can be broken into submodels based on function, subassemblies, subsystems or any means the user desires. (Note that fluid submodel division is defined differently.) For this example we will create a second submodel for the environment.
Notice in the Model Browser window that the submodels "strut" and "environment" are now listed under Thermal Submodels. If the Model Browser window is not viewable, use the main menu to select ***View>Model Browser.***	

Cryogenic Tank Support

In the upper tool bar area of the work space, notice that there are two pull down menus, one labeled "Thermal" and the other "Fluid." These pull downs show the current model. Any elements added to the network will be created in the appropriate current submodel. The pull down list can be used to change the current submodel.

A third pull down is labeled "Layer" this is the current layer network elements will be created on. When a new submodel is created a new layer will also be created. It is the users responsibility to organize elements into submodels and layers using these pull downs.

46. To create boundary nodes for the environment, click the boundary node icon. (▽)
47. Click the left mouse button adjacent to node strut.1 to create node representing the chamber wall.
48. Place a second node on the network digram to represent the radiation environment. It can be placed to the right of the column of strut nodes.
49. Place a third node adjacent to node 10 for the wall of the cryogenic tank. Place it adjacent to the bottom node (strut.10).
50. Double click the first boundary node created (chamber wall) to open the node edit form.
51. Set the temperature to **300**.
52. Click **OK** to save and exit the edit form.
53. Repeat the previous node edit step on the radiation environment node, setting the temperature to **110**.

The problem has three boundary conditions: the chamber wall, the effective vacuum environment, and the vessel wall.

Set the following boundary temperatures:

Chamber Wall	300K
Radiation Env.	110K
Tank Wall	40K

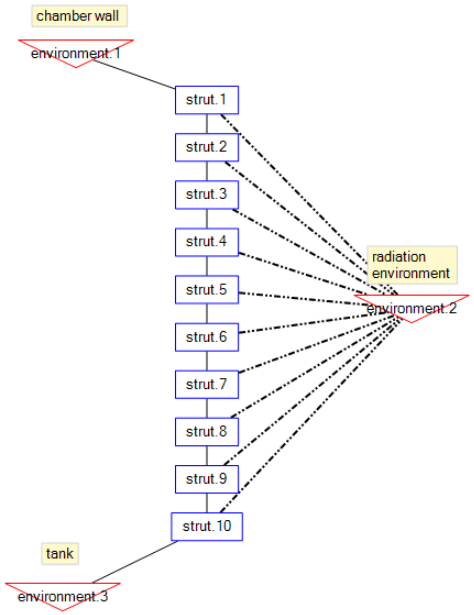
(A diagram of the model is available in the pages below if you aren't sure where to place the nodes. They can be relocated at any time.)

54. Open the Object Default window using the main menu option **Model>Defaults & Units.**
55. On the Defaults tab, change the default conductance (COND) to **2*.25*pi*diameter^2/(length/num).**
56. Click **OK** to save and exit the default dialog.
57. Draw conductors between node strut.1 and the chamber wall at the top of the diagram, and between strut.10 and the tank wall at the bottom of the diagram.

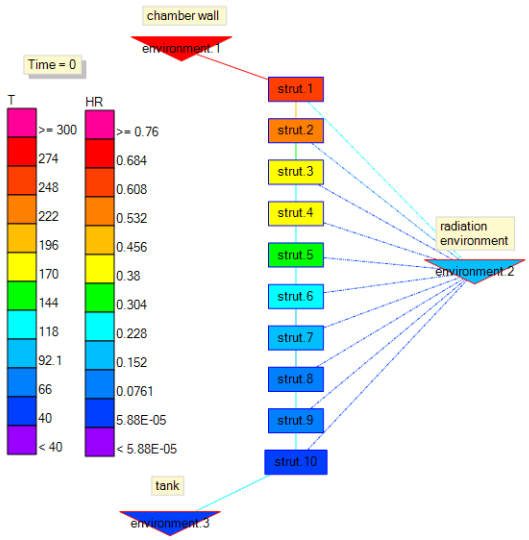
Note that the conductors at the ends of the strut have twice the conductance of the others since they represent half the length of a typical node.

The default is changed here, but as an alternative, the conductors could be added and then edited later (either separately or together).

Cryogenic Tank Support	
58. Using the network menu, select **Select>Thermal>Conductors>Plain** to select all conductors for editing. 59. Place the cursor over a conductor and right click to bring up the object edit menu and then select **Thermal...>Edit Conductors** 60. Click on the Type Specific Data tab to make the dialog active. 61. Using the pull down material menu select **strut material** 62. Click **OK** to save and exit.	Like nodal capacitances, conductors can be defined based on the Thermophysical Properties database. When a material is specified, the conductance field becomes the Area/Length multiplier: the aspect ratio. Note that only that data which has changed will be applied to the selected set. In other words, the fact that the first and last conductors have twice the aspect ratio (half the length) as the others will not be lost by changing to a specific material. Note too that the "conductance" has now be converted to an aspect ratio, and that the thermal conductivity of the strut material will be applied to this aspect ratio, yielding units of conductance of W/K in this case.
63. Click the comment icon (C). Place the cursor near the boundary node representing the chamber wall and click the left mouse button. A comment will appear. 64. Click the comment to enter edit mode and replace the text with the label **Chamber Wall**. Right-click away from the comment to end the comment (or left-click in a new area to start a new comment). 65. Add similar comments to label the tank and the radiation environment	Comments are useful for documenting a model and labeling portions of a network diagram.
Radiation conductors will be used to represent the exchange between the strut and the vacuum environment. These conductors will be defined as $G = \sigma \varepsilon A$ where σ is the Stefan-Boltzmann constant, ε is the emissivity of the surface and A is the surface area of the node. The Stefan-Boltzmann constant was previously set in step 6. The emissivity will be defined as a register named "emis."	

Cryogenic Tank Support	
66. Draw a conductor between node 1 of the strut nodes to the radiation environment node. 67. Double click on the new conductor to open the edit dialog. 68. On the Basic Data tab, **check the box labeled Radiation.** 69. Click the Type Specific Data tab to activate the type specific dialog. 70. Place the cursor in the conductance field and press the middle mouse button and then select **Edit Registers.** 71. Add a registered named **emis** and define the expression as **0.3.** 72. From the registers menu select **Exit>Save and Exit** to dismiss the register dialog. 73. At this point the conductor edit dialog should still be active. Replace the contents of the conductance field with **emis*(pi*diameter*(length/num))** 74. Click **OK** to save and exit. 75. Hover over the newly created conductor and click the right mouse button, select **Default>Set as Default**. 76. Draw new conductors between each of the strut nodes and the node representing the radiation environment.	Radiation conductors are visually distinguished from linear conductors by the dashed line. Similarly, one-way conductors are distinguished by an arrow at one end of the conductor with the arrow head representing the "downstream" end. Since network links cannot be copied, the "Set as Default" option provides an alternate means of copying links. 
The network definition is now complete. The Case Manager will be used to set up the parameters that will control the solution process. The Case Manager is comprised of three tabbed forms: Case Sets, Case Data, and Post-processing. The default case "base" will be used in this example. The Case Data tab is used to set up the solution and its control parameters. The Post-processing tab controls post-processing parameters once the case has been run.	
77. From the main menu, select **File>Save**	Save the model thus far.

Cryogenic Tank Support	
78. Open the **Case Manager** by clicking on the Case Manager tab in the left task panel or through the main menu, **View>Case Manager.** *Important: The Case Data tab provides access to general parameters, solution setup, output control, configuration build, and advanced options.* 79. Click to the Case Data tab to activate the dialog. *Important: Sinaps is defaulted to run a steady state case. The set-up can be seen by clicking on the Solution tab and then the Steady State tab. We will not change the default set-up for this case.*	Since no transient event exists, the problem can be solved using a steady-state solution. The Solution tab provides dialog to set up a steady, transient or parametric run. In each case the basic required control parameters are all defaulted. For this problem we will run the default solution of "steady." The default output will print temperatures to the text output file and everything will be saved in the binary save file or CSR folder (used for post-processing and plotting).
80. Click the Post-processing tab at the top of the Case Manager. 81. On the Thermal tab, use the Conductor pull down menu in the *Color or Plotting* area to select **Heat Rate (HR).**	By default the network diagram will automatically be post-processed for node temperatures. For this problem we are interested in the heat rate through the conductor into the tank. So we will add post-processing of conductors for heat rate.
Before solving, inputs will first be checked...	
82. Using the main menu, select **Tools>Preferences**. The preferences dialog should appear on your screen. 83. Click on the **Hover Text** tab to activate the dialog. 84. Using the data field menus, verify or select **Temperature** for Nodes, **Conductance or A/L** for conductors. 85. Click **OK** to save and exit. 86. Place the cursor over various nodes and conductors in the network. Verify temperatures and conductances are correct. (Note that the conductance A/L expression is now preceded by "15*" ... the value of the thermal conductance of "strut material.")	It is recommended to check inputs before running a model. Two modes are available, hover text and coloring the network. Hover text preferences control which input parameters will be displayed for the various network elements. If a parameter is defined as an expression, the expression will display. If the parameter is defined based on a processor variable which may not yet be defined (they are only defined while a solution is in progress), the hover text will display as N/A.

Cryogenic Tank Support	
87. Using the network menu, select **Input Checks>Color Selections.** 88. The network diagram will be colored based on hover text selections. Two color bars will appear, one for node temperature and one for conductances. They will appear on top of each other, so drag one away from the other so that both are visible. 89. Using the network menu, select **Input Checks>Reset** to reset the network.	The network can be colored based on input parameters as an alternative means of input checking. When colorized, the two conductors at the ends of the strut should have twice the conductance as the axial conductors along the rod. The radiation conductors should be very small: 0.000942.
The model will now be run using SINDA/FLUINT, a current version of which is presumed to have been installed and properly licensed (including evaluation licensing).	
90. Click the Case Sets tab of the Case Manager to activate the dialog. 91. Click **Run.** *Important: As the model runs, processes will appear on the task bar showing progress as the model is preprocessed, compiled, and launched.* 92. At the completion of the run, a window will appear stating “Successful completion of the processor”. Click **OK** to dismiss window. 93. The network will automatically post-process at the completion of the run. Color bars will appear for node temperatures and conductor heat rates. Color bars can be relocated by clicking on the color bar and holding the left mouse button down while dragging the bar to a new location. 94. Place the cursor over the conductor between the strut and the tank at the bottom of the network. The hover text will display the heat rate through the conductor. The value should be approximately 0.246 watts.	The following files/folders will be created in the base directory: • pp.out - preprocessor output file • astap.for - compiler input • base.out - text output file • messages.txt - list of messages or warnings generated during the run • base.sav - solution results for post-processing and plotting 
95. From the main menu, select **File>Save**	Save the model.

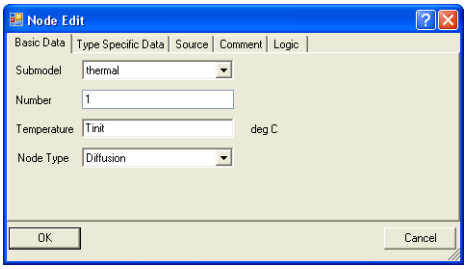
Cryogenic Tank Support
As a follow-on tutorial, try creating the above strut using a "pipe without fluid" (see "Types of Pipes" on page 7-8). Hint: a fake fluid submodel will be required. Set the pipe as "circular" with a wall definition as "standard", set OD=diameter and set thickness to diameter/2 to make a solid rod. Use a multi-conductor to connect to the radiation environment.

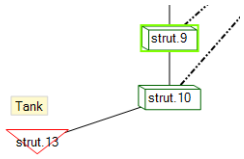
12.4 Cryogenic Tank Support - Transient Variation

Extending the previous sample, assume that a 100 watt heater, located on the rod near the vessel, is turned on at time zero. It is desired to know how long such a load can be applied before the heat leak into the vessel exceeds 1W.

A constant heat source can be applied to a node near to the tank using source data. However, because we only want the heat load to be applied during the transient event (and not during the steady state initialization that precedes it), we will control the heater by applying the source with an expression as demonstrated later.

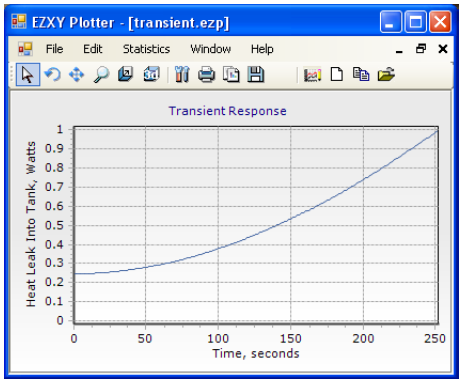
Cryogenic Transient Variation	
1. Open the basic Cryogenic Tank model by double clicking on the previously created model file **strut.xml.smdl**. This will open the model in Sinaps. *Important: If you have not done the previous tutorial, copy the files strut.xml.smdl and TdThermo.tdp from the completed tutorials directory in the Sinaps installation to your working directory.*	Start from the last version of the model and save it as a new model. The transient simulation will require the rod to be defined using diffusion nodes. The basic model will be edited and the nodes changed from arithmetic to diffusion. A new case will be added to the model for this transient simulation.

Cryogenic Transient Variation	
2. In the Model Browser window, click on **Network Diagram** to open the network window if it is not already visible. 3. If the network is post-processed (colorized), in the select the network window menu item **Post-processing>Visible** to toggle the visibility of the previous post-processing to an off state. 4. Using the network menu, click **Select>Thermal>Nodes>Arithmetic** 5. Place the cursor over any selected node and click the right mouse button to access the object edit menu. Select **Thermal...>Edit Nodes.**	There are several ways to select objects in a network. Drag-selecting from upper left to lower right selects all objects within the window. Windowing from lower right to upper left selects all objects within *and crossed by* the window. Alternatively, object selection with filtering is available through the network Select menu.
6. The Node Edit dialog will appear.  7. In the type field, use the pull down menu and select **Diffusion.** 8. Click on the Type Specific Data tab and in the material field, select **strut material.** 9. In the volume field input **.25*pi*diameter^2*length/num** 10. Click the **OK** button to save and exit.	Since we have defined a material in the Thermophysical Properties database for the strut material, we will edit the nodes to point to the database. Notice that when a material is selected on the Type Specific Data tab, the data field labeled “Capacitance” becomes “Volume.” Sinaps will automatically add the specific heat and density from the material database to the conductor. Also note that the volume is A*L: this is not the same expression that was used for conductance aspect ratio, which is A/L.

Cryogenic Transient Variation	
11. Double click on the second strut node adjacent to the tank (highlighted in figure at right: strut.9). The Node Edit dialog will appear. 12. Click on the Source tab. 13. In the source field, input **(timen>0)? load : 0** 14. With the cursor still in the source field, click the middle mouse button to access the pop up menu, select **Edit Registers.** 15. Create a new register named **load** and set it to a value of **100.** 16. Click **Exit>Save and Exit** to exit the registers dialog. 17. Click the **OK** button in the Node Edit dialog to save and exit.	A heat load will be added to the node adjacent to the tank. To make the heat load active only during a transient, we will define it using an expression based on time ("timen" means "time now" or current solution time in SINDA/FLUINT).  Conditional expressions can be used in any data field. The following expression states "if the current time (timen) is greater than 0, then set the value (heat load in this case) to 100, else set it to 0." (timen>0)? 100 : 0

Cryogenic Transient Variation	
18. Click on the Case Sets tab to make it active. Highlight the case “base” (it will turn blue). 19. Click the **Rename** button. 20. In the Rename Case window provide the name **Steady** to distinguish it from the new transient case to be added. 21. With the Steady case still highlighted, click the **Copy** button. 22. In the New Case Name window click in the data field and provide the new name **Transient**. 23. With **Transient** still highlighted, click the Case Data tab to activate the dialog. 24. Click on the Solution tab, making it activate. Click on the “Transient” tab and **check** the box for Transient (FWDBCK). 25. In the End Time field type **(environment.hr2 > 1)? timen : 1.e5** (If your conductor between the strut and the tank is labeled something other than 2, use your conductor ID number in the environment.hr# term.) *Important: Using this conditional statement for the problem endtime will initially set the endtime to a large number. When the heat rate through conductor 11 exceeds 1 watt, the problem end time will be set to the current time (timen) and the simulation will stop.*	Cases can be added, copied, renamed and deleted. The action of renaming or copying a case will automatically rename all associated files with the case. In this case, copying the case to “transient” will ensure unique input, output, and save file are created for the case to avoid conflict with the previous strut model created in the last tutorial. *Important: A steady state case must be run first to initialize the system at a condition representing a stable cold equilibrium state. To achieve this, the “Steady” solution will remain checked **and** the “Transient” solution will be added. This results in the steady case setting proper initial conditions for the subsequent transient.* *Important: When addressing processor variables such as “hr11”, the submodel name must proceed the variable in the form submodel.variable. If the reference is located within a logic block associated with the same submodel, the submodel name is optional.*
26. Click on the Output tab to activate the output dialog. 27. In the Output Time Increment data field, replace the default expression of “0.01*TIMEND” with **1** (in SI units, this is seconds).	The default output increment for a transient simulation is “0.01*TIMEND”, TIMEND being the end time for the case. In this model the duration of the event is unknown, so the output increment should be changed to a value on the order of 1 second.

Cryogenic Transient Variation	
28. Click on the Case Sets tab of the Case Manager, if Transient is not the selected case, select it and click on **Run** 29. Upon successful completion of the run, the network will post-process. The time label on the color scales will provide the time required for the heat leak into the tank to reach 1 watt.	If everything was set up and input properly, the model should come back with an answer of approximately 253 seconds for the duration of the transient.
30. Click the Post-processing tab at the top of the Case Manager 31. On the Thermal tab, use the Conductor pull down menu in the *Color or Plotting* area to select **Heat Rate (HR)**	This step was already done for the "base" case, but must be repeated for the "transient" case.
32. Click on the conductor between the strut and the cryogenic tank to highlight it. 33. Using the network windows select **Post-process>X-Y Plot** or use the icon in the post-processing toolbar.	Plot the transient heat load into the tank.
34. In the plot window, right click on the plot and select Customize (or select the icon) 35. In the top row of tabs, make sure **Chart** is selected 36. In the second row of tabs, select **Titles** 37. In the text window in the "Style" tab input field select "Plot1" and replace with **Transient Response**	Customize the plot title

Cryogenic Transient Variation	
38. In the second row of tabs, select **Axes** 39. In the list of Axes on the left, select **Left Axis.** 40. In the tabs associated with the Axes dialog, select **Title** 41. Replace “HR” with **Heat Leak into Tank, watts** 42. In the list of Axes on the left, select **Bottom Axis** 43. In the tabs associated with the Axes dialog, select **Title** 44. Replace “Time” with **Time, seconds** 45. Click **Close** to save changes 46. The plot can be saved using the EZXY plotter menu **File>Save As** and providing a name such as **transient.ezp2**	Change axis labels. The plot should appear similar to the one below. The axis resolution will vary depending on the size of the plot 
47. From the main Sinaps window menu, select **File>Save As.**	Save the model.
For more information on the usage of the EZXY Plotter, please see the EZXY Help File. Recorded webinars, availble at http://crtech.com/webinars.html, provide an introduction to the plotter usage and customization options.	

12.5 Cryogenic Tank Support - Temperature-dependent Conductivity Variation

Extending the transient cryostrut tutorial, compare the transient response if the model is built using temperature-dependent thermal conductivity. The temperature-dependent conductivity is defined as:

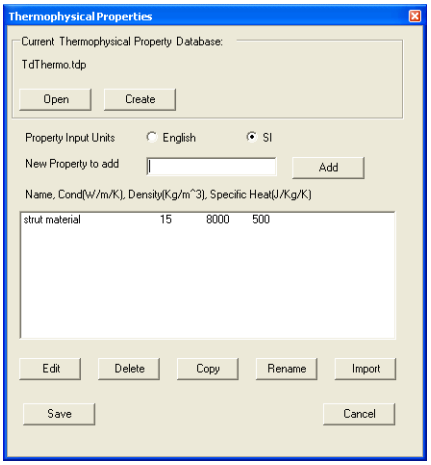
25 W/m-K @ 50K
20 W/m-K @ 100K
10 W/m-K @ 300K

The conductivity is assumed to be linear between these defined points.

Cryogenic Temperature-varying Conductivity	
1. Open the transient Cryogenic Tank model from the prior tutorial by double clicking on the model file **transient.xml.smdl**. This will open the model in Sinaps. Also click the transient.exp file, which will open EZXY in a separate session. *Important: If you have not done the previous tutorial, copy the files transient.xml.smdl and TdThermo.tdp and transient.ezp2 from the completed tutorials directory in the Sinaps installation to your working directory. Run the transient case and open the plot file.* 2. Use the menu option **File>Save** as to save the model to the new name **transientVC.xml.smdl**. 3. In the Model Browser window, click on **Network** to open the network window.	Start from the previous tutorial. Since this model will use different material properties for the support strut, we will save it as a separate model.

Cryogenic Temperature-varying Conductivity

4. Using the top-level menu, click **View>Thermophysical Properties**



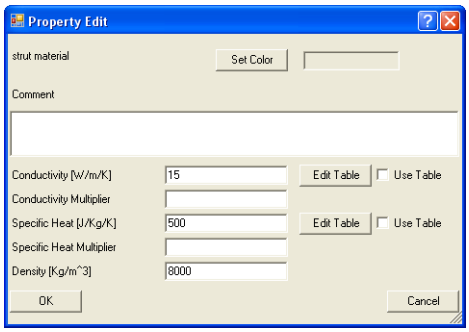
5. Click **Open** to associate a new thermopysical property file to the model.
6. In the Open window you will see TdThermo.tdp. Right click on it and select **copy**. Right click within the white space around the file list and click **paste** to create a copy of the *.tdp file.
7. Still in the Open window dialog, right click on the new file and click rename. Rename the file to **tempDependent.tdp**. Hightlight this new file and click **Open**.

The simplest approach to this problem is to modify the existing thermophysical properties database to define temperature-dependent properties or point to a new property file.

For the purpose of demonstration we will copy the existing property file to a new name, point the model to it and modify the properties to be temperature dependent.

Cryogenic Temperature-varying Conductivity

8. In the list of properties, click “strut material” to highlight it and then click the **Edit** button
9. The property edit dialog will appear.



10. Click the **Edit Table** button to the right of the conductivity data field. The array input dialog will appear
11. If not already selected, select the temperature option **K**, for degrees Kelvin.
12. In the left pane, type the following input:
    50,25
    100,20
    300,10
13. The data will be plotted on the right side of the dialog. It should appear as show at the right.
14. Select **OK**
15. Make sure the Use Table check box to the right of the conductivity field is checked.
16. Select **OK**

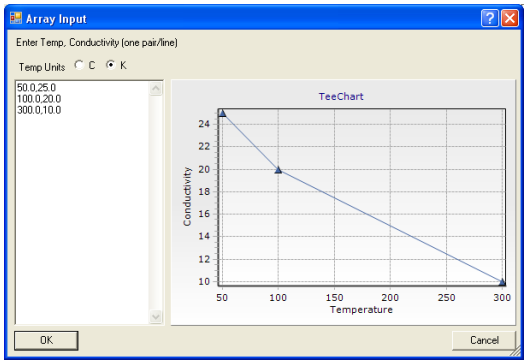
    Important: Notice that in the list of materials in the database, an “” appears next the conductivity value in the table. This indicates that the property as been defined as temperature-dependent.*
17. Select **Save** and click the **[X]** in the upper right corner of the Properties dialog to dismiss the window.

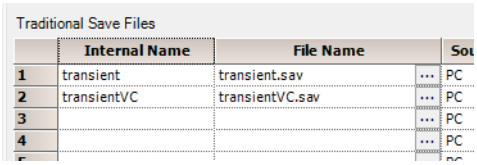
Define the conductivity as:

25 W/m-K @ 50K
20 W/m-K @ 100K
10 W/m-K @ 300K

When inputting temperature-dependent properties, the user has a choice of absolute or standard temperature units. Before inputting, data be sure to select the radio button associated with the temperature units which will be input. Once input, Sinaps will convert the data to absolute units. The radio buttons can be used to toggle between the two temperature scales.

Note that each data pair must be input on a separate line with the temperature define first in the string. Temperatures should be monotonically increasing throughout the table. If not, Sinaps will resequence the inputs during the save process.

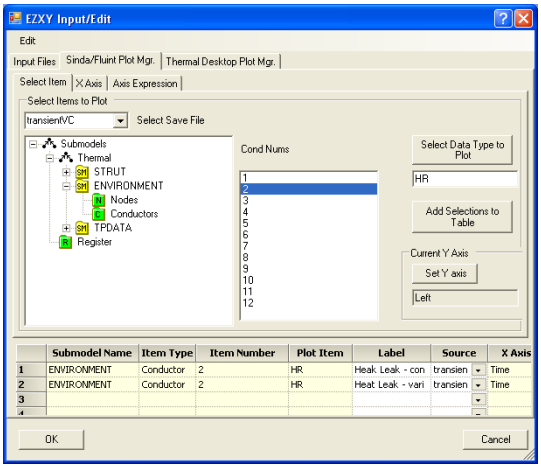


Cryogenic Temperature-varying Conductivity	
18. Click on the Case Sets tab of the Case Manager to make the dialog active. 19. Click the case **transient** to highlight it. 20. Click **Rename** and change the name to be **transientVC** 21. Click on **Run** to launch the case.	The purpose of renaming the case is to rename all the associated files, with the purpose of retaining the results of the prior run so that they can be used for comparison purposes. Also note that both "steady" and "transientVC" will run the temperature-variable property version of the model, since the property database has been modified and it applies to all cases.
22. From the main menu, select **File>Save**	Save the model.
23. In the plot window (which was opened by clicking the transient.ezp2 file), click **Edit>Add/Edit** 24. To add data to the plot, click the **Input Files** tab 25. In the lower table (Traditional Save Files), row 2, File Name column, click the "**...**" to open a browser window. 26. Browse to "transientVC.sav" in the working directory and click **Open**.	Customize the plot to compare the temperature dependent property results to the base transient. When finished, your Input Files section should look something like this (allowing for a different directory file structure on your machine):  Traditional Save Files Internal Name / File Name / Sou 1 transient transient.sav ... PC 2 transientVC transientVC.sav ... PC 3 ... PC 4 ... PC The left-most column contains an "internal name" which you can use to refer to in the Plot Manager.

Cryogenic Temperature-varying Conductivity

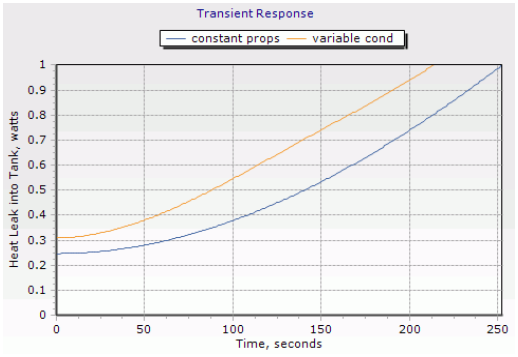
27. Click the **SINDA/FLUINT Plot Mgr.** tab
28. Use the pull down option for “Select Internal Name” and select **TransientVC**.
29. Expand the Thermal Submodel tree by clicking on Thermal.
30. Expand the ENVIRONMENT submodel tree by clicking on the + until “Conductors” appears
31. Click on **Conductors**
32. In the right pane, select **2**
33. Click the **Select Data Type to Plot** button
34. Select **Heat Rate (HR)**
35. Click **OK**
36. Click the **Add Selections to Table** button
37. In the selection table, edit the “Label” for each line. Highlight the current text and replace with **Heat Leak - constant K** for the top line and **Heat Leak - variable K** for the second line.
38. Click **OK**

Your Plot Manager tab, when finished, might look something like this:



39. Next we will customize the legend in the plot window. Right click on the plot and select **Customize**.
40. Click on the **Titles** tab below “Chart”. Provide the new title **Property Comparison**
41. Click the **Legend** tab to activate
42. Click the **Position** tab to activate
43. In the Position box, select the radio button associated with **Top**.
44. Click **Close**
45. From the plotter window menu, select **File>Save** and provide a name for the plot file.

Customize and save the plot, which should look something like this:



12.6 Cryogenic Tank Support - Parametric Sweep

Find the emissivity that minimizes the heat leak into the tank.

This can be performed one of two ways. First, a parametric sweep of the emissivity can be performed varying the emissivity from 0.0 to 1.0. After plotting the heat leak as a function of emissivity, the optimum emissivity will be easily identifiable. Alternatively, the SINDA/FLUINT Solver (optimization module) could be used to perform this analysis by setting the *object* to be the heat leak into the tank, the *goal* to -1.0e30 (to minimize the object), and setting one *design variable* as the emissivity (register EMIS). For demonstration purposes we will use the first option, a parametric sweep.

Cryogenic Strut Parametric Sweep	
1. Open the previous created base Cryogenic Tank model by double clicking on the model file **transient.xml.smdl**. This will open the model in Sinaps. *Important: If you have not done the strut tutorial, copy the files transient.xml.smdl and TdThermo.tdp from the completed tutorials directory in the Sinaps installation to your working directory.* 2. In the Model Browser window, click on **Network** to open the network window.	We can start with either the basic cryogenic strut tutorial from Section 12.3 or the transient version from Section 12.4, and modify the master model file to create a parametric case to vary the emissivity of the strut surface.
3. Use the menu **View>Case Manager** open the Case Manager dialog.	The Case Manager should contain a case call “steady” and a case called base.
4. Click the **Add** button 5. Type in a new case name **Parametric** 6. Click **OK**	We will add a second case to this model for the parametric using the Add command.
Important: The new case should be the only case highlighted in the case mangager. If it is not, click on the case name to highlight only it, thus activating the case for editing 7. Click the Case Data tab to activate the dialog 8. If not active, click the **Solution** tab and then the **Steady State** tab. 9. Uncheck the **Steady State** option	Next the case needs to be edited to set up the parametric run. We can leave the steady state solution option checked for our parametric, but this will result in Sinaps running a steady state case before the parametric, which makes an extra point on any resulting plot which can cause confusion. To keep the results clean for plotting, we will drop the initial steady state run and only run the parametric option.

Cryogenic Strut Parametric Sweep	
10. Click on the **Parametric** tab to activate the dialog. 11. Select the check box Parametric Sweep 12. Use the pull down list under Register to select the register which represents the emissivity, **EMIS** 13. Type in a lower limit of **0** 14. Type in an upper limit of **1** 15. Type in **10** for the Nsteps/Inc. 16. Under Procedure, use the pull down list to select **Steady**	Next the parametric will be defined. The parametric will vary the emissivity from 0.0 to 1.0 over 10 steps. The number of steps must be input as an integer
17. Click the **Post-processing** tab 18. Click the **Thermal** tab to make active it is not already visible. 19. Under Color or Plotting, use the Conductors pull down list to select **Heat Rate (HR)**	Set up post-processing parameters
20. Click the Case Sets tab to return to the main case dialog 21. Click the **Run** button	Run the case
22. Using the network window menu, select **View>Links Label Toggle**	To aid with directions in this tutorial, we will turn the conductor labels on.

Cryogenic Strut Parametric Sweep

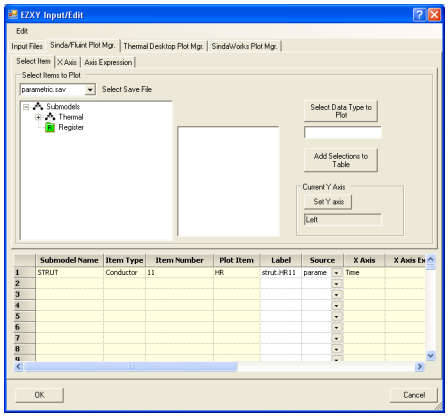
23. Click on **conductor environment.2** (between the strut and the tank) to highlight it
24. Use the network menu to select **Post-processing>X-Y Plot**
25. At the prompt “Time scale for environment.HR2 too small - setting X axis to NREC”, click **OK**.
26. At the prompt “Show this message for other curves as well?, click **No**.

    Important: These two prompts appear because EZXY recognizes there is no time data since a steady state solution has been created. Instead of time, the record numbers will be used for the x axis.
27. In the plot window, use the menu to select **Edit>Add/Edit**
28. Click the **SINDA/FLUINT Plot Mgr.** tab if it is not already visible.
29. Click the **X Axis** tab
30. In the list of X-axis options, click the “**+**” next to **Register** to expand the list
31. Select **EMIS**
32. Click the button **Set** (for all entries using this internal name)

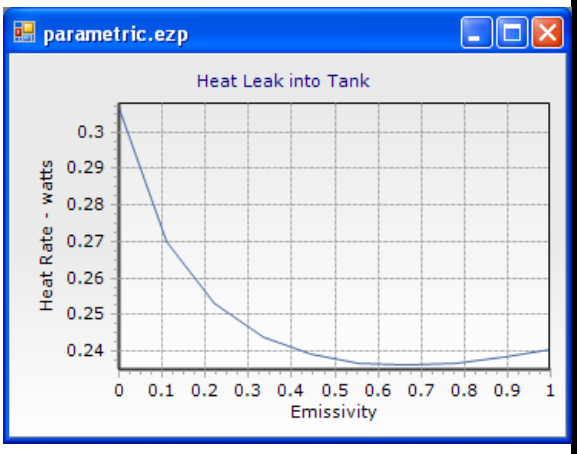
    Important: Notice in the table of items to plot, the entry in the X Axis column should now display EMIS.
33. Click **OK**

Now plot the heat leak into the tank as a function of emissivity. The heat leak is stored as the heat rate through the conductor into node 13, the boundary node representing the tank.

The plot must be edited to set the x-axis to be based on the register EMIS instead of the default of time (timen).



The plot shows a low point where the heat leak is at a minimum.



Cryogenic Strut Parametric Sweep	
34. Use the menu **View>Model Browser** to make the model browser dialog active. 35. Double click **Register** to open the registers table. 36. Click on the row number which contains the register EMIS. (Clicking the row number highlights the entry.) 37. Using the Registers menu, select **Post-process>Text** 38. In the network diagram, select **conductor environment.2** again. 39. Using the network menu, select **Post-processing>Text**	Alternatively, post-processing of the results for text values can be performed. Text post-processing of network and registers must be done separately. After post-processing the heat rate as text, the minimum heat loss can be seen in the seventh record (number 2143*). The corresponding record for the emissivity table shows that the minimum occurs at an emissivity of 0.667. *The record number of "2143" is for Version 5.3, and can change in future SINDA/FLUINT versions.
40. From the main menu, select **File>Save**	Save the model.

Important: A few words about multiple cases. The Case Manager allows several variations of a master model to be defined. In the tutorial just completed the model now contains three unique simulations all based on the same master model, a steady state, a transient, and a parametric analysis.

Multiple cases can be lauched back to back. For example highlighting all three cases in the case manager will launch all three cases. The order they are executed in is based on their order in the list of cases. Note that each case is distinct and when running two cases back to back each case will start with it's own initial conditions. A second case will not start from the final state of a prior case, however this can be done using restart options (see "Options Data" on page 9-15) or by running multiple solutions in a single case.

To run multiple solutions within a single case, select more than one solution option in the Case Manager's Solution tab. For example to run a steady state followed by a transient simulation, check the soltuion box on both the Steady and Transient solution tabs. Since these are run in the same case (or simulation), the transient solution will continue from the final state of the steady state solution. When multiple solutions are selected in the Case Manager, they will be run in the order of left to right as the solutiont tabs are displayed (steady, then transient, then parametric). To define a different order, a custom operations block must be created.

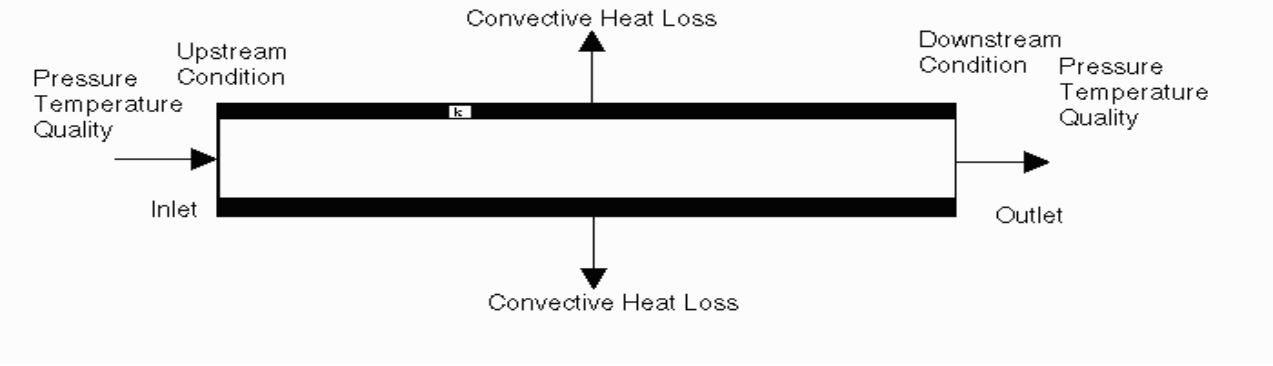
The Case Manager is a valuable tool in that you can update the network or registers in the master model and have the changes instantly propagated to all the cases. It also allows the analyst to define case specific user logic, case specific registers and overrides to global registers, table overrides, and much more. Please see "Using the Case Manager" on page 9-1 for detailed usage of the Case Manager.

12.7 Pipe Flow: Basic Fluid Flow in a Pipe

Pipe flow is common to a variety of engineering applications, such as compact heat exchangers, coolant systems, municipal water distribution systems, chemical processing, and HVAC systems. While many of these situations are commonly associated with a single-phase working fluid, two-phase flow systems are also very common.

The thermohydraulic process associated with a pipe is shown schematically in Figure 12-2. Here, a working fluid flows from an infinite source into the entrance of the pipe. As the fluid travels along the length of the pipe, the fluid pressure drops due to friction with the pipe walls, or other friction-causing devices (bends, elbows, etc.), and possibly due to acceleration forces if density gradients occur or if the flow area changes. The pressure of the working fluid many also change due to elevation changes of the pipe. The temperature (and perhaps quality or vapor/gas mass fraction) of the working may also change due to either an applied heat load or to heat transfer with the environment, and is affected by the thickness and conductivity of the pipe wall, and the flow velocity.

FIGURE 12-2. Pipe Flow Overview



The purpose of this tutorial is to demonstrate basic fluid modeling building in Sinaps, specifically chilled water flowing through an insulated horizontal pipe. Given an infinite source of fluid coming into the pipe, the goal is to find the pressure drop across the pipe and determine the exit temperature of the water.

The incoming water, at a pressure of 101.324 kPa) and a temperature of 5°C, is flowing at 0.5 kg/sec. The exit of the pipe drains to a reservoir at 101.324 kPa at 22°C. The pipe is an ANSI B36.10 steel pipe, 10 meters in length, 2 inch (50 mm) nominal diameter, schedule 40s, with a wall roughness fraction (ε/D) of 1.2e-3. Properties for steel are density of 7500 kg/m^3, conductivity of 21 W/m-C, and specific heat of 0.5 kJ/kg-K. The insulation is one inch thick with a conductivity of 0.065 W/m-C.

Variation: Determine the pressure drop across the pipe if the pipe is tilted with the outlet elevated 0.5 meters above the inlet.

Pipe Flow	
1. Copy the template file **templateSI.xml.smdl** to a new directory called pipe. 2. Rename the file **pipe.xml.smdl** 3. Open the file with Sinaps. A message will appear stating “Property file not found.” Click OK and Sinaps will create a new empty thermophysical property file.	Start this model from the previously created template based on SI units. A completed version of the template can be found in the tutorials directory located in the Sinaps install directory.
4. Open the model browser using the menu **View>Model Browser** 5. Click on the **[+]** next to fluid submodels to expand the list of fluid submodels 6. Click on the **[+]** next to the existing fluid submodel (FLOW) to expand the component list 7. Click on **Fluids**. The right panel of the browser panel will display current fluids for the submodel 8. Right click on Air(A) and select **Edit Fluids** 9. In the Define Fluids window, click on Air to highlight and then click the **Edit** button 10. In the Library field, use the pull down list to select **Water**. 11. Click **OK** 12. Click **OK**	Change working fluid from air to water. The parenthetical “(A)” refers to the species letter identifier, which is relevant only when multiple species are used to define a submodel’s working fluid. NOTE: The library water is approximate and should be replaced for any production work. A more complete/accurate description derived from NIST’s REFPROP is available at www.crtech.com Note also that the problem deals only with single-phase water, yet a two-phase water description is used. This can cause slower executions and possibly distractions (flashing, cavitation, etc.). A nonvolatile (9000 series) description is also available from www.crtech.com as a simplifying assumption.

Pipe Flow	
13. Open the thermophysical properties dialog using the main menu **View>Thermophysical Properties** or click on the icon. 14. Type in a “New Property to add” **Steel** 15. Click **Add** 16. Enter **21** as conductivity 17. Enter **500** as specific heat 18. Enter **7500** as density 19. Click **OK** 20. Type in a “New Property to add” **insulation** 21. Click **Add** 22. Enter **.065** as conductivity 23. Click **OK** 24. Click **Save** 25. Click on the **[X]** in the upper right corner to close the properties dialog	Create thermophysical properties for the wall Create a property description for the insulation on the exterior of the pipe. Since the insulation will be modeled as arithmetic nodes (negligible mass), a full property description is not necessary. The thermophysical properties window can be closed or docked. The docking may be more convenient for later access if needed.

Pipe Flow

26. Open the register table by double clicking on **Registers** in the model browser or click on the register icon ( )

Create require registers.

27. Change the existing register as shown below

Name	Expression
Tinit	22

Registers can be defined as values or expressions using built-in constants and built-in functions. To see built-in values, use the register menu **Save/Show>Show Builtin Values**.

28. Add the following registers with comments as appropriate.

Name	Expression
Tsource	5
Psource	Pinit
ALsource	ALinit
flowrate	0.5
wallrf	1.2e-3
Hconv	15

(Hint: try cutting and pasting from this table directly into the columns of the Global Registers form to save typing and potential typos.)

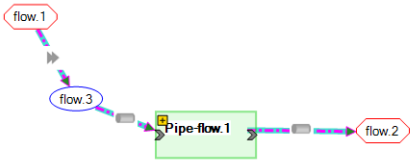
29. Using the register window menus, select **Groups>Rename Group**.
30. Enter the new name **Parameters**.
31. Using menus, select **Groups>Add Group**.

Registers can be organized into groups. These groups may be based on subsystems, component control parameters, dimensions, or any arbitrary system the user implements. If a group is deleted, all registers in the group will also be deleted. Registers can also be moved between groups.

32. In the lower part of the register window, click on the tab labeled **Registers Group 2** to move to the group display.
33. Rename this register group to **Dimensions** (see steps 29 and 30).
34. Enter the following registers in the group “dimensions”.

Name	Expression
length	10
Tinsul	1.*.0254

35. Use the register menu **Save/Show>Save Values** to save the changes.

Pipe Flow	
36. If the network diagram is not visible, open it using the menu **View>Network** or click the Network icon ( ) 37. Select the plenum icon . 38. Place the cursor in the network diagram and click the left mouse button (a lump will appear on the diagram) 39. Place a second plenum on the diagram 40. Click the junction icon and then click the cursor on the diagram to place junction 41. Click the right mouse button over a blank part of the diagram to exit "object creation" mode.	Create fluid lumps. The infinite water source upstream of the pipe can be modeled with a plenum. Since we are not interested in the transient filling of the downstream reservoir, we can assume it to be an infinite volume and model it with a second plenum. Notice the curser is a "+" during object creation mode and turns to an arrow when not in object creation mode.
42. Click the **set flow** icon or use the Tool Pallet. Hover the cursor over the first plenum (flow.1) until the link hand appears. **Drag a link** to the junction. 43. Click the **pipe** icon and then click on the network diagram to place a pipe macro on the diagram. 44. **Draw a link** from the junction to the upstream connect arrow (left side >) on the pipe. 45. **Draw a link** between the downstream connect arrow on the pipe to the second plenum created (flow.2).	A SETFLOW device will be used to force a constant flowrate of water into the pipe. The pipe will be modeled using the Sinaps PIPE component, creating both the thermal network elements for the pipe wall and the fluid elements for the water flowing through the pipe.  The links which connect the pipe to the network are owned by the pipe. Their type and parameters are all defined by the pipe form.

Pipe Flow	
46. Edit the upstream (inlet) plenum, flow.1, by double clicking on the lump's icon 47. Set the temperature to the register Tsource by highlighting the contents of the temperature field and then clicking the right mouse button. In the object menu select **Registers**. 48. In the subsequent list of registers select **Tsource**. 49. Set pressure to **Psource** 50. Set quality to **ALsource** 51. Set the lump status to **NORM**	The Lump Status field designates if the fluid is flowing or stagnant within a lump. In this case, what is happening upstream of the pipe is unknown other than fluid is flowing into the pipe, so the status of the upstream plenum will be set to be "NORM".
Important: The upstream state is specified as static, which really is the same as specifying the temperature and pressure at a flowing point. The term "infinite volume" might imply a very large, zero-velocity vessel. If this were actually the case, then the state would be better defined as "stagnant" or "total." Both situations are represented by a plenum in FLUINT, so the difference between "specify the temperature and pressure at a point in the flow" (such as a point known from measurements or controlled by some upstream mechanism), and "define an infinite, zero-velocity source" (such as an opening to/from atmospheric air or a reservoir) must be specified using the LSTAT flag. This flag is set at the bottom of the Basic Data tab of the Lump Edit form. By default, it is set to stagnant ("STAG") for plena, and to static ("NORM") for tanks and junctions.	

Pipe Flow

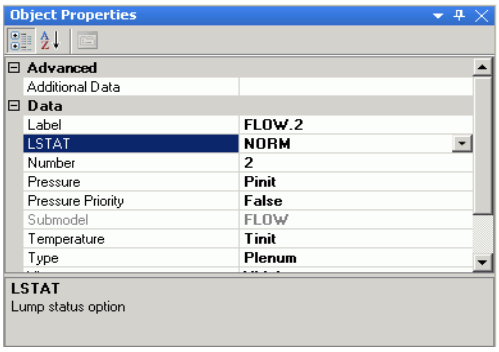
52. Open the Object Properties dialog using the main menu **View>Properties**

    Important: Please read note in the right column for important information regarding the Object properties dialog.

53. Single click on the downstream plenum (flow.2) to highlight it.

    Important: Notice that the Object Properties window has been refreshed and shows the parameters associated with the selected network object.

54. In the Object Properties window find LSTAT in the left column. Click in the associated right column and use the pull down list to change the value to NORM.



55. Close the Object Properties window by clicking on the [X] in the upper right corner.

This step demonstrates an alternate method for editing network objects through the Object Properties dialog. Advanced users may find this method of editing quicker, at least for some tasks. When working in the properties dialog, clicking the cursor in any data field will display a field description in the lower portion of the Object Properties panel.

You don't have to close the Object Properties window now. Instead, you can leave it open to see what else can be edited by that means. It can always be closed later.

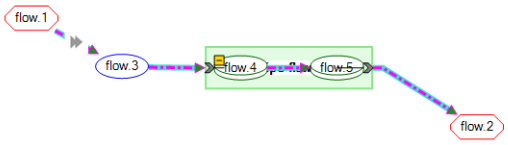
For this problem, since we are not interested in what happens after the fluid leaves the pipe, we can leave the status option on the downstream plenum set to the default of STAG. However for the purposes of teaching, we will change the lump status to NORM to demonstrate how to use the Object Properties dialog for editing a lump.

*Important: The Object Properties dialog is considered a fast method for the **advanced** user to quickly make changes to one or more objects (lumps, nodes, paths, color bars, etc.). It is also the only way to perform certain advanced changes (e.g., a lump's label, vertical/horizontal for a color bar). **The user is cautioned that input checking is not invoked when editing via the object properties dialog.** In contrast, the network element edit forms provide input checking to ensure necessary parameters are defined.*

56. Double click on the set flow link in the network to open the path edit form.
57. Click on the **Type Specific Data** tab to activate dialog
58. Set mass flow rate to maintain data field to be the register **flowrate**
59. Click **OK**

Edit SetFlow device to specify the flow rate.

Setting mass flow rate makes it a FLUINT "MFRSET" connector, while setting volumetric flow rate makes it a "VFRSET" connector.

Pipe Flow	
60. From the main menu, select **File>Save**	Save the model.
61. Click on the [+] in the upper left corner of the pipe to expand the network 62. Double click on the frame of the pipe in the Network Diagram to open the Pipe Edit dialog. *Important: A pipe is like a collection, but a closed one: one whose member elements are controlled by the program instead of arbitrarily defined by the user. A pipe is a subnetwork representing a fluid passage, or duct, with or without a 1D string of nodes and conductors representing the wall.*	Expanding the pipe will display all the network elements associated with the macro. Pipes may be viewed as expanded or collapsed. 
63. On the General tab select the Pipe Type as **Pipe with wall (HX)**. 64. Click on the Geometry tab to move to the next portion of the pipe form.	Edit the pipe. Pipes can be defined as fluid only, fluid with wall, or wall only. Notice the geometry tab provides access to storing pipes into a pipe library for later use in other models. This tab also provides a comment area useful for providing additional documentation of the pipe, perhaps and part number.

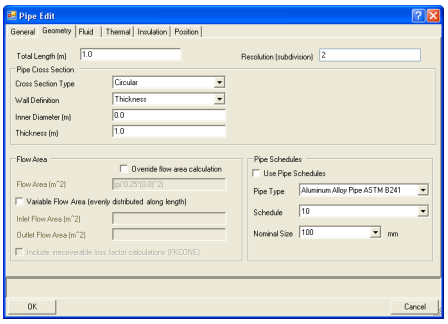
Pipe Flow

65. On the Geometry tab, set the total length as **length**. (Click the middle mouse button to access a list of registers, then select "length.")
66. Set the Resolution to be **10**
67. In the Pipe Schedules area of the form (lower right) check the box to "Use Pipe Schedules."
68. In the Pipe Type field, select **ANSI B36.10-75 & ANSI B36.19-65**.
69. For the Schedule, select **40S**
70. For the Nominal Size, select **50 mm**.

Note: When using the schedules, a pipe type reference table is first selected. Then the schedule must be defined prior to the size. Sinaps will automatically define the outer diameter and thickness fields based on the tables.

A pipe can be defined using cross sectional information or using built-in tables for standard pipe sizes, referred to as *pipe schedules*. For this tutorial the pipe schedules will be demonstrated.

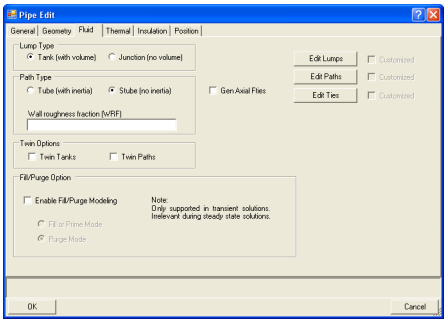
If the model is going to be used to support parametric studies on line diameter, then the pipe schedules should not be used. Instead the pipe should be defined as "circular" with a "wall definition" defined either as thickness based (ID and thickness) or standard (OD and thickness).



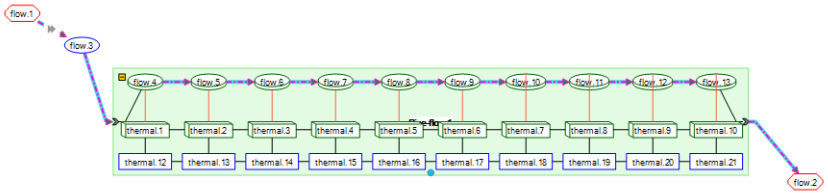
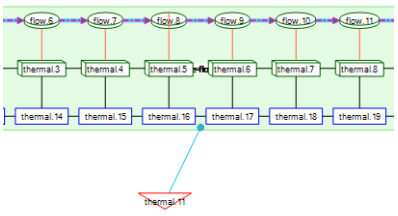
Important: Note that when the pipe schedules are used, the "Pipe Cross Section" portion of the form will become gray. Although inactive, these fields will be updated based on the pipe schedule selection.

Important: When using the Pipe Cross Section field to define a pipe, the "Cross Section Type" refers to the flow area and the "Wall Definition" defines how the wall cross section will be input. e.g. inner diameter and thickness, mass per unit length, etc.

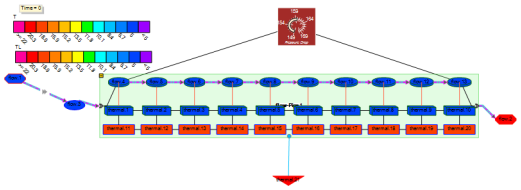
71. Move to the Fluid tab.
72. Leave the lump and path types at the default settings, tanks and stubes.
73. Define the wall roughness to the register **wallrf.**



Pipe Flow	
74. Move to the Thermal tab. 75. In the Material field use the pull down menu to select the material for the pipe wall, in this case **steel**. 76. The wall node layout should be defaulted to a single radial node. 77. Change the type of node for the wall to be **Diffusion**.	When creating pipes with a wall or creating a wall only pipe, the wall material must be specified using the materials defined in the Thermophysical Properties database. Pipes can be define with either one or two radial nodes. In some models where diffusion nodes are required for the wall, using two radial nodes, the inner node arithmetic and the outer node diffusion, will be more stable than a model with a single diffusion node.
Important: Based on the inputs defined on the geometry, fluid, and thermal tabs, Sinaps will automatically calculate lump volumes, path lengths and flow areas, thermal node volumes, and axial/radial conductances. Once a pipe is created, elements within the pipes should not be edited for these parameters or any other parameter defined by the pipe form. If a long pipe contains a reducer which changes the diameter, it should be defined as two pipes with a reducer between them. The reducer can be modeled as a (s)tube with AFI and AFJ defined to match the adjacent pipes.	
78. Move to the Insulation tab. 79. To include outside insulation on a pipe, the “Add Outside Insulation” box must be checked. Check the Add Outside Insulation box. 80. In row 1 of the table, use the pull down in the Node Type column to select the insulation node type as **Arithmetic.** 81. Use the Material pull down and select **Insulation** as the material properties. 82. In the Thickness column, define the insulation thickness as **Tinsul.** 83. Click OK to save the changes and exit the Pipe form.	Insulation can be added to the outside of the pipe through the pipe form. Sinaps will automatically create the associated thermal network based on the dimensions provide. Sinaps allows up to 10 layers of insulation to be defined, each with its own node type, material, and thickness. Note: Insulation nodes can be suppressed in the network if necessary. To do so, right click on the pipe frame and select “Insulation visibility toggle.” The visibility of the insulation nodes will be turned off and the pipe background will turn pink as a visual reminder that the pipe has insulation defined.

Pipe Flow	
84. Click on the **[+]** in the upper left corner of the pipe to expand the subnetwork. You may have to move the expanded pipe so that it doesn't cover the outlet plenum, or you can move the outlet plenum as shown at the right. To move an icon, drag it with the left mouse button to a new location.	The diagram should look similar to the graphic below. The lumps are the oval icons. The wall nodes are displayed below the lumps and the insulation nodes below the wall nodes. Layout can be changed by right clicking on the pipe frame and selecting "Axis mirror." The layout will flip to have the insulation on top and the fluid lumps on the bottom.
	
85. Select **File>Save** to save this model.	
86. In the network window, click on the boundary node icon in the thermal toolbar, or use the Tool Pallet 87. Place the cursor in the network and click the left mouse button to place the node on the network diagram below the pipe.	Create a boundary condition for the pipe wall
88. Hover the cursor over the new boundary node until the link hand appears. Draw a line to the conductor connect port (blue dot) on the frame of the pipe. 89. Double click on the new multiconductor (blue link) to edit the conductor. 90. Click on the Type Specific Data tab to make it active. 91. In the Conductance field, type in **Hconv**. This is the convection coefficient defined in the registers. 92. Make sure the "Use Pipe Area" box is checked. If unchecked, check the box. 93. Click **OK**.	Generate conductors between each of the outermost pipe nodes and the boundary condition using the multi-conductor tool. Pipes have a blue dot on the side representing a thermal connect port which when used, generates a fan of conductors from any single node to all the outermost pipe nodes (wall if no insulation, insulation if present). Generated conductors will not display individually in the thermal network but are available in the save file for plotting and test post-processing. 

Pipe Flow	
94. From the main menu, select **File>Save**	Save the model.
95. Open the case manager if it is not visible (**View>Case Manager**) 96. Click the **Case Data** tab and then click **Output** 97. Click on the **Text** tab 98. On the **Thermal** tab, select **Temperature**, **Conductance**, and **Register Summary** 99. On the **Fluid** tab, select **Lump Data**, **Path Data**, and **Tie Data**	Set up the run information in the Case Manager. The default steady state simulation will be run, but expanded text output is desired (in addition to the standard SAVE file that will contain all data for post-processing).
100. Click on the **Post-processing** tab 101. Under **Thermal** select **Temperature** for Nodes. (Should be already set.) 102. Under **Fluid**, select **Temperature** for lumps. (Should be already set, as “TL”) 103. Check to make sure the box labeled Post-process after SINDA/FLUINT Run is checked.	Specify post-processing parameters To goal is to find the pressure drop so the network could be post-processed for lump pressure. However a post-processing gauge will be used to provide the pressure drop so the lumps will be post-processed for temperature.
104. Return to the network diagram, **View>Network** 105. Open the Tool Pallet if it is not visible, **View>Tool Pallet** 106. Click on the **Post-processing** tab 107. Click on the **gauge** ( ) and **drag** it onto the network 108. **Create links** between the gauge and the first and last lumps of the pipe macro 109. **Double click on the gauge** to bring up the edit dialog. 110. Select Function of **Delta** 111. Enter the label **Pressure Drop** 112. Click the Post-process Value button and select **Basic>Pressure (PL)** 113. Click **OK**	Add a post-processing gauge to monitor pressure drop across the pipe. Post-processing tools will be automatically scaled based on value. The automatic scaling can be overridden using the minimum, maximum and scale options in the edit form. Further customization of tick marks etc. can be performed by editing the post-processing tool in object properties.

Pipe Flow	
114.Open the case manager, **View>Case Manager** 115.Click on the **Case Sets** tab to activate 116.Click the case “base” to highlight it 117.Click **Run**	Run the case.
118.Return to the network window. The color scales can be moved to a new location. 119.Left click on one of the color scales 120.Hold the shift key and select the second color scale. Both color scales should be highlighted at this point. 121.Right click over one of the color scales and select **Synchronize scales**. 122.Click on the node temperature (“T”) scale. The object properties dialog will appear if not already visible. Change the “Orientation” to **Horizontal.** 123.Repeat for the lump temperature scale. 124.Hover the cursor over the gauge, the pressure drop across the pipe will display in the hover text., ~160 pascals 125.However the cursor over the last lump of the pipe, the temperature will display as hover text, ~5.0 degrees C	Since both color scales are temperature, they can be synchronized to show the same color scale. The orientation of a scale can also be changed to be horizontal. 
126.From the main menu, select **File>Save**	Save the model.
Variation: Determine the pressure drop across the pipe if the pipe is tilted with the outlet elevated 0.5 meters above the inlet.	
127.Open the Global Registers by clicking on the register icon ( ). 128.On the register group “Dimensions” add the register **Elevation** and set it to **0.5**.	Create one new register elevation.

Pipe Flow	
129.Double click on the exit plenum. 130.Click on the **Position** tab to activate the dialog. 131.Enter **Elevation** as the Z Axis Position value. 132.Click **OK** 133.Double click the pipe to open the edit dialog 134.Click on the **Position** tab to activate the dialog 135.Enter **Elevation** as the Vector End Point Z value. 136.Check the **Use Position Data** box. 137.Click **OK**	Modify the pipe such that the outlet is elevated 0.5 meters above the inlet. This involves raising the elevation of the outlet state, and "tilting" the Pipe by setting its outlet position (the positions of the lumps along the pipe will be linearly apportioned according to the end locations). Note: Two options are provided for pipe positioning, Inlet/Outlet Lumps and Inlet Point with Direction Vector. The inlet with direction vector is the preferred method, since it captures the elevation change associated with the pipe inlet and outlet paths. It also properly adjusts the elevations based on pipe discretization (center, upstream, or downstream).
To enable the modeling of gravitation effects on the fluid, a gravitational acceleration must be defined in Global Control Data. SINDA/FLUINT allows gravity (or any 3D body force-inducing acceleration) to be defined in each of three axis. When modeling gravity but be sure to define each lump's elevation in that same coordinate. This example uses the Z axis for gravity. *Sinaps provides a set of built-in constants and conversions. The gravitational acceleration is on of the built-in constants, GRAVSI.*	
138., Move to the Case Manager, **Case Sets** tab. 139.Click on the case "Base" and select the **Copy** button to create a new case. 140.Provide a new case name of **Elevated** 141.Click on the new case new case name to highlight it. 142.Click on the **Case Data** tab. 143.Click on the **Advanced** tab. 144.Double click on **Global Control Data** to open the dialog. 145.In the ACCELZ field enter **9.8** or click the middle mouse button in the field and insert the built-in constant **GRAVSI**. 146.Click **OK**	Create a new case the for gravity environment. Note the only differences between the two cases is the gravitational term set in the Global Control Data for the elevated case.

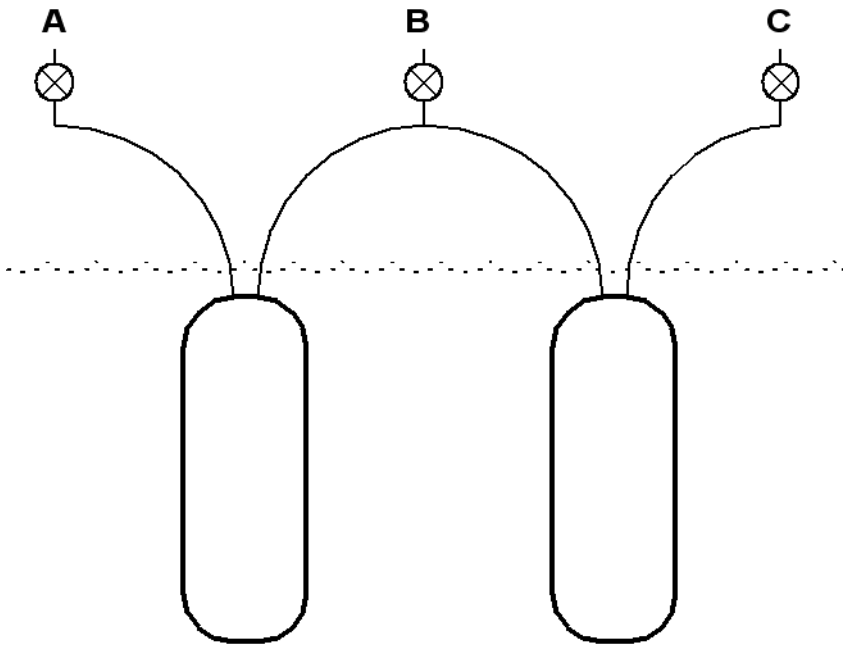
Pipe Flow	
147.Click on the **Case Sets** tab of the case manager 148.Click **Run** 149.Double click on the gauge. 150.Set the scaling factor as **0.001** 151.Click the post-process icon	Run the case. The pressure drop will automatically scale the pressure gauge, probably with a factor of 0.1, resulting in ~507 on the scale, representing 5068 Pascals. Setting the scale factor to 0.001 converts from the standard SI unit of Pascals to kilopascals resulting in a value of 5 kPa.
152.From the main menu, select **File>Save**	Save the model.

12.8 Filling Scuba Tanks

Problem Description

Consider the filling (pressurization) of two cylindrical bottles from a source of air at 22 bar (22.0e5 Pascals, or just less than 22 atmospheres) gauge pressure and 20°C. The cylinders are 10cm diameter and 60cm long, are initially full of atmospheric air, and are submerged in a large water bath also at 20°C. The heat transfer coefficient between the air and water (through the bottle walls) is 10 W/m^2-K.

Both bottles have two fill ports which are connected to 1 meter long by 1 cm inner diameter lines. The ends of these lines can either be connected to the pressure source or to each other. Shut-off valves exist to turn on or off the flow from the pressure source. When open, the valves have a flow area of 1.0e-5 m^2 and a K factor of 25 corresponding to that flow area. Below is a schematic of this configuration. The pressure source can be applied at either point A, B, or C in that diagram.



Assuming that the objective is to pressurize the bottles in a short amount of time, which location (A, B, or C) would result in the most mass being injected into the bottles by the time pressures equilibrate?

12.8.1 Model Description

The problem can be solved entirely by a transient solution, starting from stagnant initial conditions: zero flow rates, and pressures zero (gauge) everywhere.*

Lumps and Paths. The pressure source will be treated as a plenum: an infinite source or sink or mass and energy at a constant pressure and temperature. By assuming that the bottles are well-mixed, a single FLUINT tank will be used to represent each of the bottles. CTLVLV connectors will be used to represent the valves, and STUBE connectors the lines. Junctions are then needed to represent dead-ends in unused valves (say, B and C if A is connected to the pressure source) and to join series connectors. By using junctions for these points, the volume of the lines and valves is itself neglected as small compared to that of the bottles.

Choking might initially occur within the valves. Choking is checked automatically by FLUINT, but for the sake of simplicity that topic will not be covered in this tutorial.

Nodes and Ties. The water bath is a thermal boundary condition: it will be treated as a SINDA boundary node in submodel "water." It will be connected to the bottles using HTU (user) ties, whose UA is the given heat transfer coefficient times the surface area of each bottle.

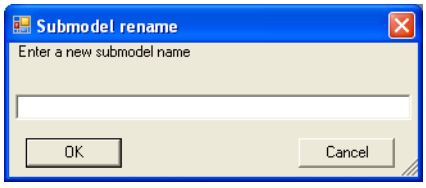
* This sample problem is a rare exception: in general it is a bad idea to start a FLUINT transient solution without first performing a steady state (FASTIC) run to start from realistic initial conditions.

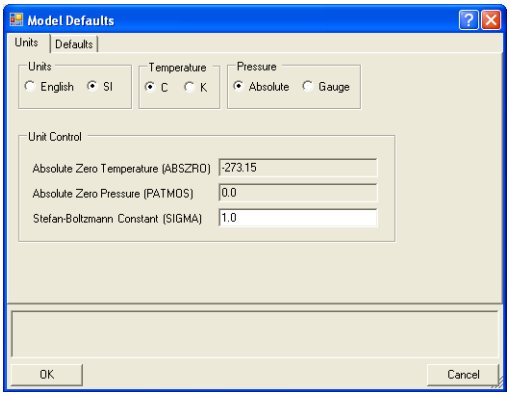
Solution Sequence. In this particular sample problem, a single transient run is needed. Such solutions are requested by calling single routines such as:

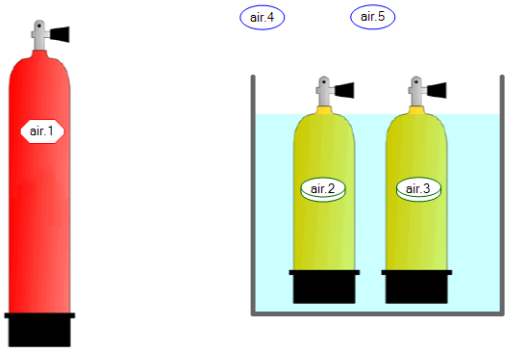
```
CALL TRANSIENT
```

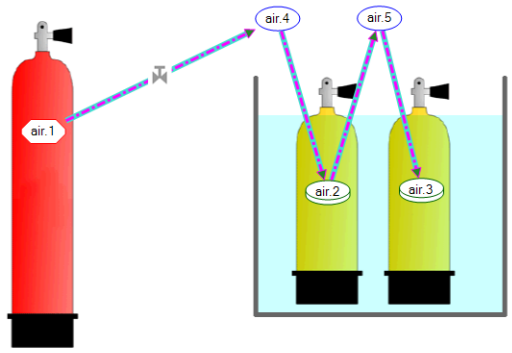
Since the run time is unknown, the TIMEND (end time) parameter will be set to stop the run when the pressure of a bottle reaches 99.9% of that of the source. This can be easily accomplished using the conditional statement below. An output interval of 1 second is chosen

```
air.pl2 > 0.999*air.pl1)? timen : 1.0e30.
```

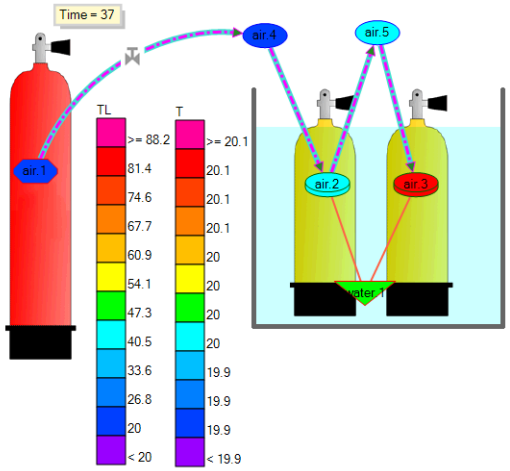
Filling Scuba Tanks	
1. Open Sinaps by double clicking on the template created in an earlier tutorial (templateSI.xml.smdl). 2. Click the menu **File>Save As** to save the template as a new name in a new folder.	Based on the data provided we will use the SI system for units. The units in the template are SI, degrees C, and absolute pressure.
3. Use the menu option **Model>Defaults & Units** to open the *Model Defaults* dialog. On the units tab, change the pressure option to **Gauge.**	We need to change the pressure to "gauge." Note, when Gauge is selected the PATMOS field is automatically updated.
4. Open the Model Browser by clicking on the **Model Browser** tab in the left panel. If not present, use the menu option **View>Model Browser**. 5. Expand the thermal submodel list by clicking on the [+] to the left of "Thermal Submodels". 6. Click on the submodel name "thermal" to highlight it. 7. Select the menu option **Model>Rename**. The following window will appear. Type in **Water** and click **OK**.  8. The Find Replace window will appear next. See notes to the right herein and then click **Cancel** to dismiss. 9. Repeat the above process to rename the fluid submodel to **Air**.	We will need both a thermal and fluid submodel for this problem. Our template already has a thermal submodel named "thermal" and a fluid submodel named "flow. We can change the name of a submodel to be more descriptive. In this case we will rename the thermal submodel to water and the fluid submodel to air. When a submodel name is changed, the Find Replace dialog will automatically appear. This dialog will search the entire master model for instances of the model name and propagate the changes. In the template file we started with, there is no logic or tables so we can skip this step.

Filling Scuba Tanks	
10. Open the Global Registers by clicking on the register icon ( ).	Delete old registers and import new registers
11. Delete the registers **Pinit**, **Tinit** and **ALinit** by highlighting the rows (click on the row numbers to select) and hit the delete key to remove the rows from the table.	Holding the control key while selecting rows allows multiple rows to be selected. Holding the shift key allows a block of rows to be selected.
12. In the register window, select the menu option **Save/Show>Import Registers.**	Import registers from a file.
13. Browse to and select the file **registers.xml** in the Sinaps install directory / tutorials/scuba/.	
14. Click **Open** to complete the import process.	
15. In the register window select **Save/Show>Save Values** to save the new registers.	
1. Open the *Model Defaults* window again and click on the **Defaults** tab. 	
2. For Node Temp enter **Twater**	Use the middle click option in any field to access a list a registers to quick paste into a field.
3. For Diam enter **Dhose**	
4. For Length enter **Lhose**	Set the lump defaults to be applicable to the tanks
5. For Lump Temp enter **Twater**	
6. For Vol enter **VolBot**	Set the path defaults to be applicable to the tubing.
7. For Void Fraction enter **0**	
8. Click **OK** to save and close the window.	

Filling Scuba Tanks	
9. **Right click** anywhere on the network diagram background. The **Object Properties** dialog should automatically open (lower left corner of window) if it is not already open. 10. Click in the field labeled **Background Image**. A browse button (...) will appear at the right side of the field. Click it to open a browse window and browse to the file **tanks.gif** in the Sinaps install directory / tutorials/scuba/. 11. Click the **Zoom Out** icon ( ) to allow the full image to be seen (may need to click a few times).	Insert a background picture. Background graphics will always be placed in the upper left of the network diagram and cannot be resized once placed. However graphics can be resized, modified or collages made in a graphics packages outside of Sinaps and then used as a background image. Zoom options are useful for working with large networks. Options include zoom in or out, zoom extents, or overview (see "Network View Options" on page 4-13 for details).
12. Drop a plena ( )on the large red source tank in the graphic. 13. Drop one tank ( )on each of the yellow scuba tanks. 14. Drop two junctions ( ) above the scuba tanks to represent the two fill valve options. 	The pressure source will be treated as a plenum: an infinite source or sink or mass and energy at a constant pressure and temperature. By assuming that the bottles are well-mixed, a single FLUINT tank will be used to represent each of the bottles. Junctions are then needed to represent dead-ends in unused valves (say, B and C if A is connected to the pressure source) and to join series connectors. By using junctions for these points, the *volume* of the lines and valves is itself neglected as small compared to that of the bottles. Note that connecting the source to A or C is effectively the same system so we only need to model A and B.
15. Edit the source plena by double clicking on it. 16. Set the temperature to **Tsource** and pressure to **Psource**.	Since the fluid is air, we only need to define pressure and temperature. The void fraction field can be ignored.

Filling Scuba Tanks	
17. Click the control valve ( ) path icon and draw a link between the source lump and the first tank. 18. Click the STUBE icon ( ) and draw paths between the junctions and the tanks as shown below. 	Paths are drawn in the direction of flow with the arrow indicating direction. Reverse flow will have a negative flowrate. CTLVLV connectors will be used to represent the valves, and STUBE connectors the lines. Choking might initially occur within the valves. Choking is modeled by default in FLUINT although it can be disabled by setting the Choked flow method (MCH) parameter of a path to "No detect (0)".
19. Double click on the control valve path to edit it. 20. In the **Basic Data** tab of the edit form, set the Flow Area to **Avalve**. 21. On the **Type Specific Data** tab, set the K factor to **kvalve**.	Since we previously set the path diameter and length defaults to match the hose, we do not need to edit the STUBEs, however the Control Valve will need to be edited to set the flow area of the valve and the K-factor.

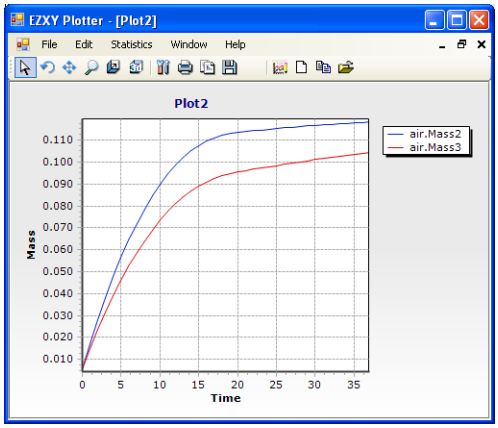
Filling Scuba Tanks	
22. Click the boundary node icon (▽) and place a node on the network to represent the water bath. 23. Draw a link between the node and the left tank to create a TIE (note that a tie will automatically be created since it is the only type of network element which can connect a node to a lump). 24. Double click on the new tie to edit it. In the Tie Type field, use the pull down and select **User Described Heat Transfer (HTU)**. 25. Move to the **Type Specific Data** tab and enter **UAwater** in the **Conductance of the tie field**. 26. Right click on the new tie and select **Default>Set as Default**. 27. Create a new tie between the node and the second tank.	The water bath is a thermal boundary condition: it will be treated as a SINDA boundary node in submodel "water." It will be connected to the bottles using HTU (user) ties, whose UA is the given heat transfer coefficient times the surface area of each bottle. Ties are used to describe heat flow between a thermal and fluid submodel. Ties are always "owned" by the fluid submodel so if the fluid submodel is not "built" in the Case Manager, the tie will not be active when a case is run. Ties can be calculated by SINDA/FLUINT during the run (convection, HTN*), user defined (HTU*), or pool boiling (HTP). See "Heat Transfer Ties" on page 5-101 for more information on ties.
28. Click the menu **File>Save** to save the model.	The network is complete. Save the model to prevent loss of data.
29. Open the Case Manager (click on tab in left panel or use the menu option View>Case Manager). 30. Click the case **base** to highlight it, and then click the **Rename** button. 31. Enter the new name **Side Fill** and click **OK**. 32. Click the **Case Data** tab to activate the dialog.	Set up a case for the side fill option. We can rename the case "base" to be more descriptive. When a case is renamed all associated files (input, output, save files etc.) are all renamed. The case name is limited to 72 alphanumeric characters, underscores and spaces are allowed.
33. On the Solution>Steady State tab, **uncheck t**he Steady State option. 34. On the **Transient** tab, check the **Transient** option. 35. For the End Time enter the conditional statement **(air.pl2 > 0.999*air.pl1)? timen : 1.0e30**	In this particular sample problem, a single transient run is needed. Since we do not know how long it will take to fill the tanks, instead of entering an endtime for the solution we can use a conditional statement. Set the end time arbitrarily high and then stop the run when the pressure of a bottle reaches 99.9% of that of the source.

Filling Scuba Tanks	
36. On the **Output** tab, set the Output Time Increment to **1** (every second). 37. Within the Text tab, use the Fluid tab to also select **Path Data**, **Tie Data, Choked Flow** and **Register Summary**.	Define what type of output is to be created during the run. It is up to the user to define what type of data is to be written the text based output file. In this case it would be helpful to include path data, tie data, choked flow information, and a register summary to the default thermal temperature and fluid lump tab summary. For the Save File, leave the default of "all". The save file is a binary file used for postprocessing and plotting.
38. Click the menu **File>Save** to save the model.	The case definition is complete. Save the model to prevent loss of data.
39. Click on the **Case Sets** tab of Case Manager. 40. Click on the **Run** button.	Run the model During the run, a SINDA/FLUINT Run Status window will pop up. When complete the window should state "Successful completion of the processor.
41. The network should automatically post-process upon completion. If it does not, make sure nothing is selected in the network window and click on the postprocessing icon ( ). It takes about 37 seconds for the pressures to stabilize.	Default post-processing of the network displays temperatures. 

Filling Scuba Tanks

42. Open the **Case Manager** and go to the **Postprocessing** tab.
43. On the Save File tab, Fluid tab, use the Select button to the right of Lumps to select **Tanks>Mass**.
44. Click the **Refresh** button on the post-processing tab or click the post-processing icon ( )to refresh the color display.
45. Window select the two tank lumps and click the plot icon ( ) to plot the mass of the two tanks.
46. To change the left axis format, right click on the plot and select **Customize**.
47. Click the **Axes** tab, select **Left Axis** and then select the **Labels** tab.
48. Click the **Format** tab and using the pull down to see the various options or type a new option in the window such as **0.000** which will display 3 significant digits.
49. Keep the plot window open and return to the main Sinaps window.

For this problem we are interested in which configuration provides the most mass of air in the tanks when the pressures stabilize. Creating a plot of the mass of both tanks would be beneficial. Note the junctions and plenum do not have an associated mass term.



The results show that first (upstream) tank ends up with roughly 15% more air in it compared to the downstream tank.

To model the center fill option the control valve can simply be moved to the other junction and the case rerun. However for the purposes of demonstration, for this tutorial we will create a second case for the center fill and use a register as a "switch" to control how the air supply is connected.

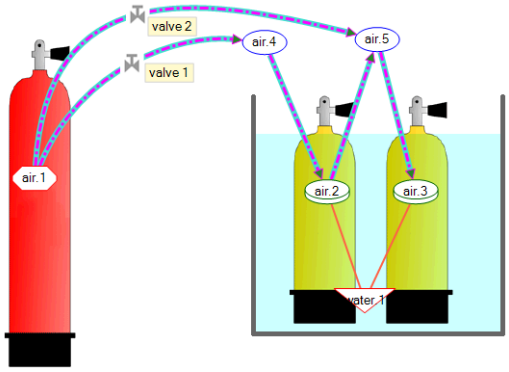
50. Open the **Register** dialog.
51. Add a new register called **switch** and set its value to **1**. Add the following comment **1 = valve 1, 2 = valve 2**
52. Use the Type pull down option to set the register Switch to be **Int** (integer).
53. Click the register menu option **Save/ Show>Save Values**.

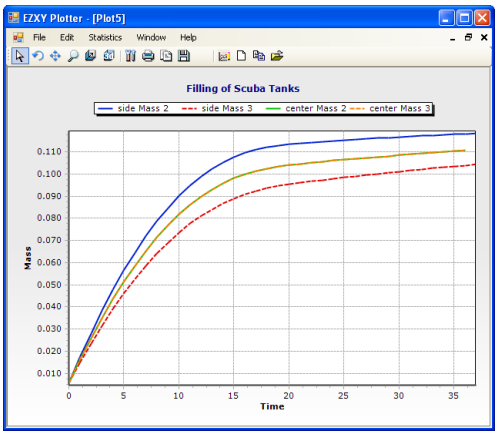
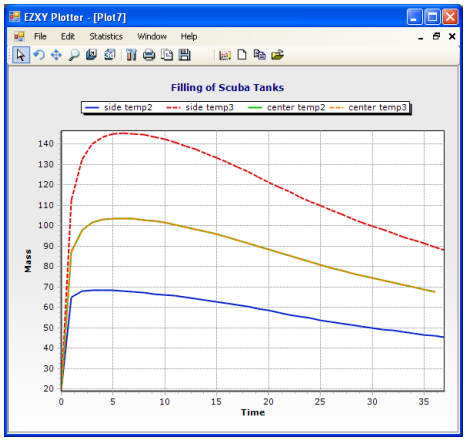
Create a new register to control the valve states. If set to the value of 1, flow will go through the first valve, if set to 2 it will go through the second valve.

When using a register as flag or switch, the register should be defined as an integer.

54. Return to the network diagram.
55. Edit the control valve by double clicking on the path icon.
56. On the Type Specific Data tab, for the K factor enter the following conditional statement
(switch==1)? kvalve : -1.0

Edit the valve and set the K-factor field to a conditional statement which sets the factor to the register KVALVE if the condition is met (switch = =1) or to a negative value if the condition is not met. The negative value will cause the control valve to shut.

Filling Scuba Tanks	
57. Right click on the control valve path and select **Default>Set as default**. 58. Draw a new path between the supply tank and the other junction (air.5) 59. Double click on the new path to edit it. 60. On the Type Specific Data tab change the K factor field to be the following (switch==**2**)? kvalve : -1.0 61. Add comment next to the values for documentation purpose by clicking on the comment icon (c) and dropping a comment field onto the network. Click the filed to edit it. Add labels such as **valve 1** and **valve 2**.	Create a second valve between the source tank and the junction air.5. 
62. Open the **Case Manager** and go to the **Case Set** tab. 63. Highlight the **Side Fill** case name and then click the **Copy** to create a copy of the case. 64. In the New Case Name dialog enter the new name **Center Fill**.	Create a second case in the Case Manager.
65. Highlight the new case and then move to the **Case Data** tab, and then to the **Advanced** tab. 66. Click on **Registers** to access the case specific register dialog. 67. In the Name column enter **switch**. Select **Int** as the type, and enter the value of **2**. 68. Click the register menu option **Exit/Save and Exit**.	Create a case specific register override for the new case to set the value of switch to "2" resulting in valve 1 being closed and valve 2 being open. When creating a case specific register, if the register name provided is an existing register, Sinaps will change the row color to blue to indicate the entry is a override of a global register.
69. Return to the Case Manager dialog, **Case Sets** tab. 70. Select both cases by clicking on one case name, hold the shift key and select the second case name (both should now be highlighted). 71. Click the **Run** button.	Multiple cases can be launched simultaneously within Sinaps. In this situation, Sinaps will launch the first case and when it completes it will launch the next case. Models are launched in the order they are listed within the case manager. Note that each case is run as a discrete run, i.e. runs are not appended. To create a run which starts where a previous run has left off, the restart option must be used.

Filling Scuba Tanks	
72. Return to the plot created in set 45. Click the Add/Edit icon ( ) to access the **EZXY Input/Edit** dialog. Select the **Input Files** tab, then click the browse button ( ) on the second row. Browse to and select the file **center fill.sav** in the working directory. 73. Click Open to add the save file to the plot.	Edit the plot to compare save files. Using the *Input Files* tab data from multiple save files can be displayed in a single plot.
74. Click on the **Sinda/Fluint Plot Mgr.** tab. 75. In the **Select Save File** field, use the pull down to select **center fill**. 76. In the model tree, expand [+] the **Fluid** submodel list, expand the submodel **air,** then click on **Lumps**. 77. As list of lump IDs will appear on to the right. Select **2** and **3** (holding the sift key for multi select). 78. Click the **Select Data Type to Plot** button. Expand **Tanks** and select **Mass.** 79. Click OK to dismiss Lump Value window. 80. Click the **Add Selections to Table** button. 81. Click **OK** to update the plot.	For more information on customizing plots including setting line color, type, legend location, background colors, etc. please see Chapter 5 of the EZXY User's Manual. The plot below has been customized. 
82. Click the Add/Edit icon ( ) to access the **EZXY Input/Edit** dialog. 83. To change the data value being plotted, use the model tree, expand [+] the **Fluid** submodel list, expand the submodel **air,** then click on **Lumps**. 84. Notice the **Select Data Type to Plot** field is defaulted to "Temperature". Click the **Change Plot Items in Grid** to change all plot items to temperature. 85. Manually update the label fields in the grid. 86. Click **OK** to update the plot.	Change the plot to look at temperatures. 

Filling Scuba Tanks

As the results show, the left tank is colder in the side fill case, and right tank is hotter since it is receiving its air from the left tank which is heating up due to compression rather than from a 20°C source as in the symmetric center fill case. This means that left tank will end up with more mass than in the symmetric case, and the right tank will with less. If the goal of the pressurization is total mass in both bottles, then either attachment point will do. But if the goal is evenly distribution mass between the bottles, then center fill option (location B) should be chosen.

For those readers curious about this effect, in which the temperature of the center fill case appears to split into higher and lower temperatures in the side fill case, the trend continues with more and more bottles in a row: the first bottle gets progressively colder, approaching the inlet condition of 20°C, while the last bottle gets progressively hotter.

APPENDIX A

Heritage Turbomachinery

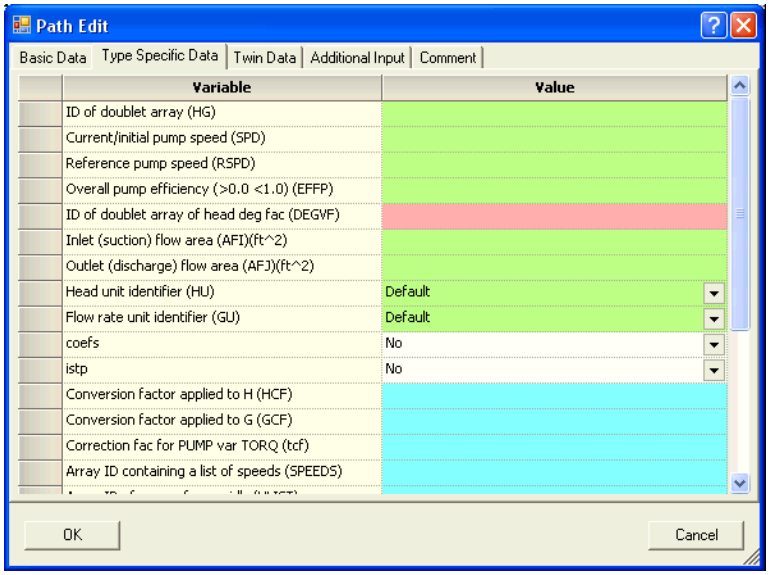
This addendum provides documentation for heritage type specific inputs for the pump, turbine, and compressors prior to the redesign of these device edit forms with version 5.4.

A.1 Heritage Pump/Fan Form (PUMP)

Figure A-1 shows the heritage (prior to version 5.4) type-specific edit form for a pump. Descriptions of each input are described below. For detailed information on describing pumps, cavitation detection and simulation, torque, heating, and other advanced options, please refer to section 3.5.13 of the SINDA/FLUINT User's Manual.

The data fields highlighted in green indicate required or commonly used parameters, blue indicates less common options and red indicates advanced options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. Most parameters may be defined using registers or expressions, excepting array IDs.

FIGURE A-1. Pump/Fan Type Specific Data Form



ID of doublet array, H vs. G (HG) - ID of doublet array (defined in the same submodel as the pump) containing a single pump head-flow curve: head (H: units according to HU) as a function of flow rate (G: units according to GU). G must increase monotonically, and no two adjacent H values may be equal (i.e., the slope dH/dG cannot be zero). The array pointed to must contain real values or expressions of the form G1,H1, G2,H2 ... or if COEFS=YES then Φ1,Ψ1, Φ2,Ψ2 ...

Initial pump speed (SPD) - This is the current/initial pump speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency.

Reference pump speed (RSPD) - reference pump speed, in consistent user units (same as SPD). This option is only applicable if the HG= method is used and COEFS=NO. If the AEFFP= method is used as well, its G values will be scaled with |SPD/RSPD|.

Overall pump efficiency (EFFP) - Pump efficiency as a single user defined value. Pump efficiencies are optional but should be specified or at least estimated (typical values are between 0.6 and 0.8, for example). Efficiencies are used to calculate heating and torque.

ID of doublet array for head degradation factor (DEGVF) - The ID of a user defined doublet array characterizing the head degradation factor as a function of void fraction at the pump inlet when two-phase is present. By default a built-in curve for DEG_{α} is used. This curve represents a reasonable average degradation for several pumps reported in the literature and can be used as a first approximation unless specific pump characteristics are known. To disable this correction (perhaps in favor of cavitation corrections), specify a DEGVF array with zero degradation values (two such points are required as a minimum to define a line).

Inlet flow area (AFI) - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Head unit identifier (HU) - Integer flag for head units: see Table 1. These units will apply to H values, but if COEFS=YES then HPMP and all array head values will be nondimensional or at least quasi-nondimensional according to HPMP = Ψ = HCF*H/SPD^2 (where H is given by HU units), and DGDH is in units of GPMP/ HPMP.

TABLE 1. Head Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

HU	Meaning	Cautions
0	default: m if uid=si, ft if uid=eng	
1	m	
2	cm	
3	mm	
4	cmH_2O, 4°C	pressure units
5	mmH_2O, 4°C	pressure units
6	mmHg, 0°C (Torr)	pressure units
7	Pa	pressure units
8	kPa	pressure units
9	Bar	pressure units
101	ft	
102	in	
103	inH_20 (39.2°F)	pressure units
104	inHg (32°F)	pressure units
105	psi	pressure units
106	psf	pressure units
201	(absolute) pressure ratio	TABULAR, TURBINE, COMPRESS, or COMPPD only
202	Martin Beta Factor	TABULAR only

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 2. These units will apply to G values, but if COEFS=YES then GPMP and all array flow values will be nondimensional or at least quasi-nondimensional according to GPMP = Φ = GCF*G/SPD (where G is given by GU units).

TABLE 2. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
0	default: m^3/s if uid=si, ft^3/hr if uid=eng	
1	m^3/s	
2	m^3/min	
3	m^3/hr	
4	L/s	

TABLE 2. Flow Units for PUMP, TABULAR, TURBINE, COMPRESS, COMPPD Connectors

GU	Meaning	Cautions
5	L/min	
6	L/hr	
7	cm^3/s	
8	cm^3/min	
9	cm^3/hr	
10	mm^3/s	
11	mm^3/min	
12	mm^3/hr	
13	kg/s	mass flow rate units
14	kg/min	mass flow rate units
15	kg/hr	mass flow rate units
101	ft^3/s (CFS)	
102	ft^3/min (CFM)	
103	ft^3/hr (CFH)	
104	gal/s (US GPS)	
105	gal/min (US GPM)	
106	gal/hr (US GPH)	
107	in^3/s	
108	in^3/min	
109	in^3/hr	
110	lb_m/s	mass flow rate units
111	lb_m/min	mass flow rate units
112	lb_m/hr	mass flow rate units

Coefficient-based Performance (COEFS) - Option to use coefficient-based performance parameters. If COEFS=NO then dimensional head and flow rates (H and G) are used to define performance parameters. If COEFS=YES, then H is interpreted as head coefficient Ψ instead of head or pressure drop, and G is interpreted as flow coefficient Φ instead of mass or volumetric flow rate. This has an effect not only on the H/Ψ and G/Φ values, but also on HPMP, GPMP, DGDH, and TCF (but *not* SSCF, ANPSH, RNPSH, ANSS, nor RNSS).

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then head is defined on the basis of total or stagnation pressures, and either AF or both AFI and AFJ are required inputs. Otherwise (ISTP=NO or equivalently, ISTP=SS), static inlet/outlet pressures are used without accounting for dynamic head.

Conversion factor applied to head (HCF) - If COEFS=YES, this is the conversion factor applied to H/SPD^2 to yield Ψ: $\Psi = HCF*H/SPD^2$. For US Customary units, with H in feet and SPD in rpm, then $HCF = g/D^2$ where D is the rotor diameter and g is the acceleration of gravity.

Conversion factor applied to flowrate (GFC) - If COEFS=YES, this is the conversion factor applied to G/SPD to yield Φ: Φ= GCF*G/SPD. For US Customary units, with G in gpm and SPD in rpm, then either GCF = 1/(A*D) or perhaps $GCF=1/D^3$ where D is the rotor diameter and A is the flow area.

Pump Torque Coefficient (TCF) - Correction factor for the PUMP output variable TORQ, such that TORQ = TCF*GPMP*HPMP/(EFFP*SPD) where GPMP and HPMP are in units defined by COEFS, GU, and HU. TCF is input by the user to perform whatever unit conversions are necessary. TCF defaults to unity.

Array ID containing a list of speeds (SPEEDS) - A full map alternative to the single-curve "HG= method," SPEEDS is the array ID containing a list of speeds for which head and flow rate curves (and possibly efficiencies) will be specified via the HLIST, GLIST, and optionally the ELIST arrays. Like HG, SPEEDS is an array ID, and the entries in that array must be real numbers or expressions in the same units as SPD.

Array ID of array of array ID's (HLIST) - The ID of a SINDA/FLUINT array (in the same submodel as the pump device) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing head values that correspond to each of the speeds in the SPEEDS array. In other words, the first array ID in the HLIST array should contain a series of head values at the first speed in the SPEEDS array. Those values should correspond to the first array of GLIST (and optionally, ELIST): all such arrays must be the same length. The array values themselves should be in units consistent with COEFS and HU.

Array ID of array of array ID's (GLIST) - The ID of a SINDA/FLUINT array (in this submodel) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing head values that correspond to each of the speeds in the SPEEDS array. In other words, the first array ID in the GLIST array should contain a series of flow rate values at the first speed in the SPEEDS array. Those values should correspond to the first array of HLIST (and optionally, ELIST): all such arrays must be the same length. The array values themselves should be in units consistent with COEFS and GU. If COEFS=YES and only one array of flow coefficient points is needed, specify that array repeatedly for the number of speeds supplied. For example: GLIST = 102, and in ARRAY DATA: 102 = 34, 34, 34, 34 ....

Array ID of array of array ID's (ELIST) - The optional ID of a SINDA/FLUINT array (in this submodel) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing pump hydraulic efficiencies that correspond to each of the speeds in the SPEEDS array. In other words, the first array ID in the ELIST array should contain a series of efficiency values (between zero and one) at the first speed in SPEEDS. Those values should correspond to the first array of HLIST and the first array of GLIST. Edge values are used if the provided range is exceeded; no extrapolations are made for this variable.

ID of doublet array of efficiency (AEFFP) - ID of the doublet array containing the efficiency as a function of flow G (or flow coefficient Φ if COEFS=YES). End values will be used if the provided range is exceeded; no extrapolations are made for this variable. If RSPD has been specified (above), then G will be scaled with |SPD/RSPD|.

Required net positive suction head (RHPSH) - This is the required NPSH in units defined by GU but not by COEFS (it is H not Φ). Can be compared with output variable ANPSH, perhaps using CHKFLASH.

Required suction specific speed (RNSS) - Required Nss (suction specific speed) in units defined by SSCF. Can be compared with output variable ANSS, perhaps using CHKFLASH.

NSS units correction factor (SSCF) - NSS (suction specific speed) units correction factor, such that ANSS= SSCF*SPD*$G^{1/2}$/$ANPSH^{3/4}$ where the flow G and the available NPSH (output variable ANPSH) are defined by GU and HU but *not* by COEFS: they are G and H, not Φ and Ψ.

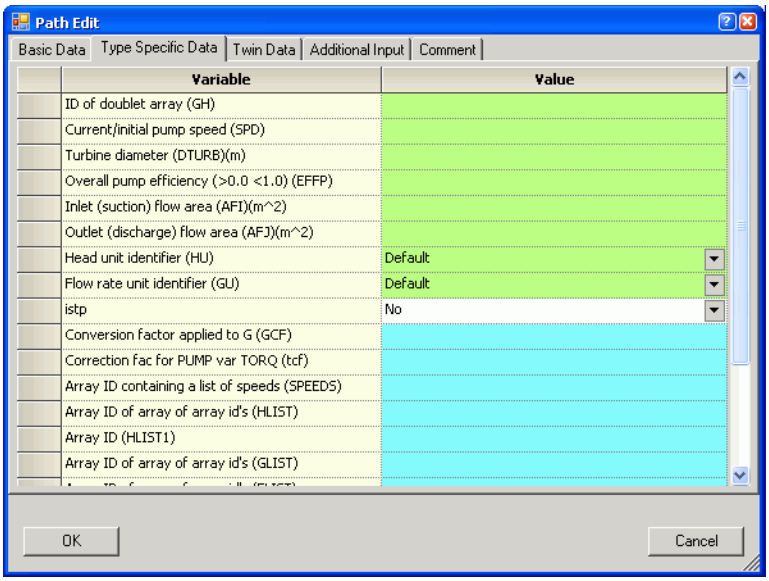
Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density. Note that ANSS etc. are always based on current inlet (suction) properties.

A.2 Heritage Turbine Device Form

Figure A-2 shows the heritage (prior to version 5.4) type-specific edit form for a turbine. Descriptions of each input are described below. For detailed information on describing turbines, choking detection and simulation, torque, heating, and other advanced options, please refer to section 3.5.15 of the SINDA/FLUINT User's Manual.

The data fields highlighted in green indicate required or commonly used parameters, blue indicates less common options and red indicates advanced options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. Most parameters may be defined using registers or expressions, excepting array IDs.

FIGURE A-2. Turbine Type Specific Data Form



ID of doublet array, G vs. H (GH) - ID of doublet array (defined in the same submodel as the turbine) containing a single turbine head-flow curve: G (flow rate: units according to GU) as a function of pressure ratio or difference (units according to HU). H must increase monotonically. *No two adjacent H values may be equal (i.e., the slope dH/dG cannot be zero: the inverse of this slope, DGDH, cannot be infinite).* The array pointed to must contain real values or expressions of the form H1,G1, H2,G2 ...

Current/Initial Speed (SPD) - Current/initial turbine speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency. Note that SPD is in actual units (e.g., rpm) and is not dependent on the value of NSPD, unlike the values in the SPEEDS array. NOTE: To use the ELISTR or AEFFP options, or for the output variable UCR (velocity ratio U/C) to be valid, SPD *must* be in units of rpm.

Turbine Diameter (DTURB) - Turbine diameter (in feet or meters, depending on UID). This input is required if either the AEFFP= or the ELISTR= method is used to define efficiency based on velocity ratio (U/C). If DTURB is input, SPD should be in units of RPM such that the output variable UCR (the velocity ratio U/C) can be used, whether or not AEFFP or ELISTR is used.

Overall turbine efficiency (EFFP) - Turbine efficiency as a single user defined value. Efficiencies are optional but should be specified or at least estimated (typical values are between 0.6 and 0.8, for example). Efficiencies are used to calculate heating and torque.

Inlet flow area (AFI) - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Head unit identifier (HU) - Integer flag for head units: see Table 1. These units will apply to H values.

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 2. These units will apply to G values.

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then pressures and other state variables are defined on the basis of total or stagnation conditions, and either AF or both AFI and AFJ are required inputs. By default ISTP=TT, and so both inlet and outlet states are based on total (stagnation) conditions and therefore flow areas (AF or both AFI and AFJ) are required. ISTP=TS (total-static) and ISTP=ST (static-total) are also available as options. ISTP also affects the meaning of TREF and PREF.

Conversion factor applied to flow rate (GFC) - A G conversion factor (defaulting to unity) that is active if either NAF or MREF is nonzero. *GCF is ignored if MREF=NAF=0.*

Turbine Torque Coefficient (TCF) - Correction factor for the TURBINE output variable TORQ.

Array ID containing a list of speeds (SPEEDS) - A full map alternative to the single-curve "GH= method," SPEEDS is the array ID containing a list of speeds for which pressure and flow curves (and possibly efficiencies) will be specified via the HLIST (or HLIST1), GLIST, and optionally the ELIST arrays. Like GH, SPEEDS is an array ID, and the entries in that array must be real numbers or expressions in the same units as SPD if NSPD=0, or equivalent speeds if NSPD=1.

Array ID of array of array ID's (HLIST) - If SPEEDS is used for a full map (the "GLIST= method"), this is the ID of a SINDA/FLUINT array (in the same submodel as the turbine) containing the values of pressure ratio H (or head or pressure drop, per HU) for which table values in GLIST correspond. HLIST is the ID of a SINDA/FLUINT array (in this submodel) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing pressure ratio (H) values that correspond to each of the N speeds set in the SPEEDS input. HLIST is a list of lists. In other words, the first array ID in the HLIST array should contain a series of pressure ratio values at the first speed in the

SPEEDS array. The first array listed should also correspond to the first array of GLIST (and optionally, ELIST), and the first value in the first HLIST array should correspond to the first value in the first GLIST array, and so forth. All such arrays within HLIST and GLIST (and ELIST or PLIST, if used), should be the same length. H must increase monotonically within each HLIST array. *No two adjacent H values may be equal.*

Alternate pressure/head ID's (HLIST1) - If SPEEDS is used for a full map (the "GLIST= method"), and if only one set of pressures ratios H (or head or pressure drop, per HU) are to be used for all GLIST arrays, then this is the ID of a SINDA/FLUINT array (in the same submodel as the turbine) containing the single list of values of pressure ratios. The first *value* listed in the HLIST1 array should correspond to the first values in each *array* of GLIST (and optionally, ELIST or PLIST), and so forth. HLIST1 must be large enough to cover all of the pressure ratios needed for the longest GLIST (or ELIST or PLIST or ELISTR) array. H must increase monotonically. *No two adjacent H values may be equal.* HLIST1 may be used in conjunction with ELISTR, even if GLIST is *not* input, or even if GLIST *is* input along with HLIST (in which case HLIST values are used instead of HLIST1 values for the pressure/flow relationship).

Array ID of array of array ID's (GLIST) - The ID of a SINDA/FLUINT array (in the same submodel as the turbine) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing flow values that correspond to each of the N speeds set in the SPEEDS input. GLIST is a list of lists. In other words, the first array ID in the GLIST array should contain a series of flow rate values at the first speed in the SPEEDS array.
The array values themselves should be in units consistent with GU. Supplied G values may be adjusted per the rules invoked via the MREF and NAF variables, and if so they might be scaled by the parameter GCF

Array ID of array of array ID's (ELIST) - The optional ID of a SINDA/FLUINT array (in the same submodel as the turbine) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing turbine isentropic efficiencies that correspond to the N speeds set in the SPEEDS input. ELIST is a list of lists. In other words, the first array ID in the ELIST array should contain a series of efficiency values (between zero and one) at the first speed in SPEEDS. The values in that first *array* should correspond to the first *array* of GLIST. Edge values are used if the provided range is exceeded; no extrapolations are made for this variable.

ID of doublet array of efficiency (AEFFP) - ID of the doublet array containing the efficiency as a function of flow velocity ratio U/C, where U is the tangential tip velocity (based on SPD and DTURB) and C is the isentropic spouting velocity for a gas turbine (fluid jet speed for a hydraulic turbine). AEFFP is therefore of the form $(U/C)_1$, η_1, $(U/C)_2$, η_2 .... where velocity ratios are monotonically increasing (i.e., $(U/C)_2 > (U/C)_1$). End values will be used if the provided range is exceeded; no extrapolations are made for this variable. If this option is used, SPD is assumed to be in rpm.

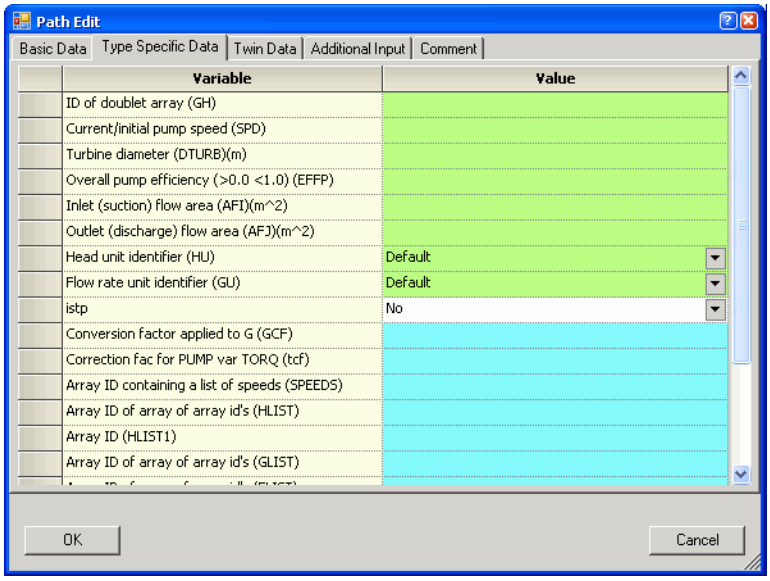
Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density.

A.3 Heritage Variable Displacement Compressor Form

Figure A-3 shows the heritage (prior to version 5.4) type-specific edit form for a COMPRESS device. Descriptions of each input are described below. For detailed information on describing compressors, surge detection and simulation, torque, heating, and other advanced options, please refer to section 3.5.16 of the SINDA/FLUINT User's Manual.

The data fields highlighted in green indicate required or commonly used parameters, blue indicates less common options and red indicates advanced options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. Most parameters may be defined using registers or expressions, excepting array IDs.

FIGURE A-3. Compressor Type Specific Data Form



ID of doublet array, H vs. G (HG) - ID of doublet array (in the same submodel as the compressor) containing a single compressor head-flow curve: pressure ratio or difference (units according to HU) as a function of G (flow rate: units according to GU). H need not decrease monotonically, but *no two adjacent H values may be equal (i.e., the slope dH/dG cannot be zero: the inverse of this slope, DGDH, cannot be infinite).* The array pointed to must contain real values or expressions of the form G1,H1, G2,H2 ...

Current/Initial Speed (SPD) - Current/initial compressor speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency. Note that SPD is in actual units (e.g., rpm) and is not dependent on the value of NSPD, unlike the values in the SPEEDS array. SPD should be specified even if using the GCOMP= method, since SPD is used for TORQ calculations.

Overall turbine efficiency (EFFP) - Overall compressor isentropic efficiency (real, between 0.0 and 1.0). This is an output if either ELIST, PLIST, or AEFFP arrays are provided. The efficiency basis is a function of ISTP, and therefore may be total-total or total-static, for example.

Inlet flow area (AFI) - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Head unit identifier (HU) - Integer flag for head units: see Table 1. These units will apply to H values.

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 2. These units will apply to G values.

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then pressures and other state variables are defined on the basis of total or stagnation conditions, and either AF or both AFI and AFJ are required inputs. By default ISTP=TT, and so both inlet and outlet states are based on total (stagnation) conditions and therefore flow areas (AF or both AFI and AFJ) are required. ISTP=TS (total-static) and ISTP=ST (static-total) are also available as options. ISTP also affects the meaning of TREF and PREF.

Conversion factor applied to flow rate (GFC) - A G conversion factor (defaulting to unity) that is active if either NAF or MREF is nonzero. *GCF is ignored if MREF=NAF=0.*

Compressor Torque Coefficient (TCF) - Correction factor for the COMPRESS output variable TORQ (see main text).

Array ID containing a list of speeds (SPEEDS) - A full map alternative to the single-curve "HG= method," SPEEDS is the array ID containing a list of speeds for which pressure and flow curves (and possibly efficiencies) will be specified via the HLIST, GLIST, and optionally the ELIST or PLIST arrays. Like HG, SPEEDS is an array ID, and the entries in that array must be real numbers or expressions in the same units as SPD if NSPD=0, or equivalent speeds if NSPD=1. SPEEDS is also a required input if either the GSURGE or the HSURGE option is used.

Array ID of array of array ID's (HLIST) - If SPEEDS is used for a full map (the "HLIST= method"), this is the ID of a SINDA/FLUINT array (in the same submodel as this compressor) containing the values of pressure ratio H (or head or pressure drop, per HU) for which table values in GLIST correspond. HLIST is the ID of a SINDA/FLUINT array (in this submodel) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing pressure ratio (H) values that correspond to each of the N speeds set in the SPEEDS input. HLIST is a list of lists. In other words, the first array ID in the HLIST array should contain a series of pressure ratio values at the first speed in the SPEEDS array. The first array listed should also correspond to the first array of GLIST (and optionally, ELIST), and the first value in the first HLIST array should correspond to the first value in the first GLIST array, and so forth. All such arrays within HLIST and GLIST (and ELIST or PLIST, if used), should be the same length. H need not decrease monotonically, but *no two adjacent H values may be equal.*

Array ID of array of array ID's (GLIST) - The ID of a SINDA/FLUINT array (in the same submodel as this compressor) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing flow values that correspond to each of the N speeds set in the SPEEDS input. GLIST is a list of lists. In other words, the first array ID in the GLIST array should contain a series of flow rate values at the first speed in the SPEEDS array. *GLIST values within each array must increase monotonically.* The array values themselves should be in units consistent with GU. Supplied G values may be adjusted per the rules invoked via the MREF and NAF variables, and if so they might be scaled by the parameter GCF.

Array ID of array of array ID's (ELIST) - The optional ID of a SINDA/FLUINT array (n the same submodel as this compressor) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing compressor isentropic efficiencies that correspond to the N speeds set in the SPEEDS input. ELIST is a list of lists. In other words, the first array ID in the ELIST array should contain a series of efficiency values (between zero and one) at the first speed in SPEEDS. The values in that first *array* should correspond to the first *array* of GLIST. Edge values are used if the provided range is exceeded; no extrapolations are made for this variable.

ID of doublet array of efficiency (AEFFP) - ID of the doublet array containing the isentropic efficiency as a function of flow rate G. AEFFP is of the form G_1, η_1, G_2, η_2 .... where flow rates are monotonically increasing (i.e., $G_2 > G_1$).

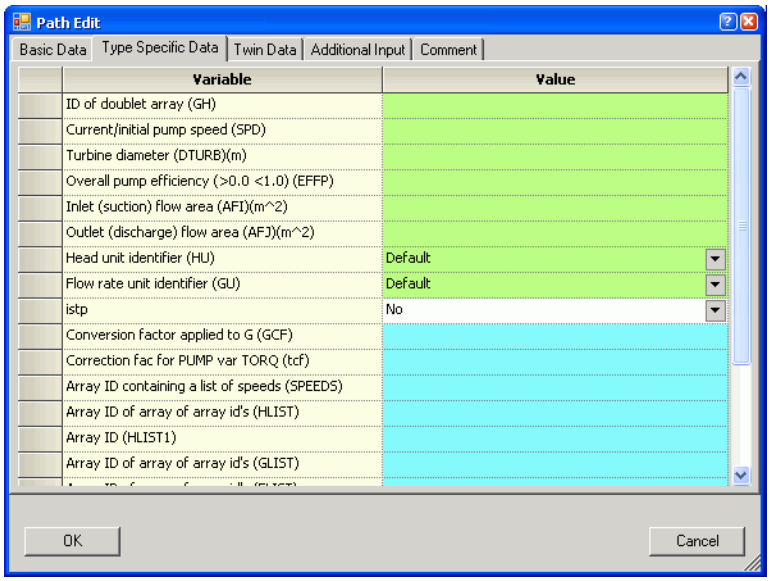
Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density.

A.4 Heritage Positive Displacement Compressor Form

Figure A-4 shows the heritage (prior to version 5.4) type-specific edit form for a COMPPD device. Descriptions of each input are described below. For detailed information on describing compressors, torque, heating, and other advanced options, please refer to section 3.5.17 of the SINDA/FLUINT User's Manual.

The data fields highlighted in green indicate required or commonly used parameters, blue indicates less common options and red indicates advanced options. Unless stated otherwise, all parameters are available in the user logic blocks and may be altered if necessary. Most parameters may be defined using registers or expressions, excepting array IDs.

FIGURE A-4. COMPPD Type Specific Data Form



ID of doublet array (EFFV) - ID of the doublet array containing the *volumetric* efficiency as a function of the actual (or equivalent) speed N. AEFFV is of the form N_1, η_{v1}, N_2, η_{v2} .... where speeds are monotonically increasing (i.e., $N_2 > N_1$). End values will be used if the provided range is exceeded; no extrapolations are made for this variable.

Current/Initial Speed (SPD) - Current/initial compressor speed, in consistent user units. Avoid very small or zero values (the default!) unless the model/inputs are specifically designed for this case. In most cases, zero SPD is equivalent to a closed valve for consistency. SPD is used for TORQ calculations.

Volumetric displacement (DISP) - Volumetric displacement of the compressor, usually in units of volume (per rotation), but any units can be used as long as the resulting G is consistent with the inputs GCF, GU, SPD, and EFFV (which is normally unitless).

Overall compressor efficiency (EFFP) - Overall compressor isentropic efficiency (real, between 0.0 and 1.0). This is an output if either ELIST, PLIST, or AEFFP arrays are provided. The efficiency basis is a function of ISTP, and therefore may be total-total or total-static, for example.

Inlet flow area (AFI) - inlet (suction) flow area, if not constant. If AFI is specified, AFJ must also be specified, and AF will be calculated automatically

Outlet flow area (AFJ) - outlet (discharge) flow area, if not constant

Head unit identifier (HU) - Integer flag for head units: see Table 1. These units will apply to H values.

Flowrate unit identifier (GU) - Integer flag for flow rate units: see Table 2. These units will apply to G values.

Total/stagnation Pressure Flag (ISTP) - If ISTP=YES (or equivalently, ISTP=TT), then pressures and other state variables are defined on the basis of total or stagnation conditions, and either AF or both AFI and AFJ are required inputs. By default ISTP=TT, and so both inlet and outlet states are based on total (stagnation) conditions and therefore flow areas (AF or both AFI and AFJ) are required. ISTP=TS (total-static) and ISTP=ST (static-total) are also available as options.

Conversion factor applied to flow rate (GFC) - A G conversion or scaling factor (defaulting to unity) which is applied to the calculated flow rate: GCOMP = GCF*EFFV*DISP*SPD.

Compressor Torque Coefficient (TCF) - Correction factor for the COMPPD output variable TORQ (see main text).

Array ID containing a list of speeds (SPEEDS) - A full map alternative to the single-curve "AEFFV= method," SPEEDS is the array ID containing a list of speeds for which pressure and flow curves (and possibly efficiencies) will be specified via the HLIST, ELISTV, and optionally the ELIST or PLIST arrays. SPEEDS is an array ID, and the entries in that array must be real numbers or expressions in the same units as SPD.

Array ID of array of array ID's (HLIST) - If SPEEDS is used for a full map (the "ELISTV= method"), this is the ID of a SINDA/FLUINT array (in this submodel) containing the values of pressure ratio H (or head or pressure drop, per HU) for which table values in ELISTV correspond. HLIST is the ID of a SINDA/FLUINT array (in the same submodel as this COMPPD) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing pressure ratio (H) values that correspond to each of the N speeds set in the SPEEDS input. HLIST is a list of lists. In other words, the first array ID in the HLIST array should contain a series of pressure ratio values at the first speed in the SPEEDS array. The first array listed should also correspond to the first array of ELISTV (and optionally, ELIST), and the first value in the first HLIST array should correspond to the first value in the first ELISTV array, and so forth. All such arrays within HLIST and ELISTV (and ELIST or PLIST, if used), should be the same length. *H values must increase monotonically, and no two adjacent H values may be equal.*

Array ID of array of array ID's (ELISTV) - The ID of a SINDA/FLUINT array (in the same submodel as this COMPPD) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing *volumetric* efficiency values that correspond to each of the N speeds set in the SPEEDS input. ELISTV is a list of lists. In other

words, the first array ID in the ELISTV array should contain a series of volumetric efficiency values at the first speed in the SPEEDS array. The array values themselves are normally unitless, but otherwise should be in units consistent with GU, GCF, SPD, and DISP.

Array ID of array of array ID's (ELIST) - The optional ID of a SINDA/FLUINT array (in the same submodel as this COMPPD) which itself contains a list of N IDs of SINDA/FLUINT arrays (also in this submodel) containing compressor *isentropic* efficiencies that correspond to the N speeds set in the SPEEDS input. ELIST is a list of lists. In other words, the first array ID in the ELIST array should contain a series of isentropic efficiency values (between zero and one) at the first speed in SPEEDS. The values in that first *array* should correspond to the first *array* of ELISTV. Edge values are used if the provided range is exceeded; no extrapolations are made for this variable

ID of doublet array of efficiency (AEFFP) - ID of the doublet array containing the *isentropic* efficiency as a function of the actual (or equivalent) speed N. AEFFP is of the form N_1, η_{p1}, N_2, η_{p2} .... where speeds are monotonically increasing (i.e., $N_2 > N_1$). End values will be used if the provided range is exceeded; no extrapolations are made for this variable.

Weighting factor for properties (UPF) - weighting factor for properties: 1.0 = base H and G on inlet (suction) density, 0.0 = base H and G on outlet (discharge) density.

Index

A

Aggregation. See also Automatic Ducts
Arrays
  Character Array 9-32
  See Tables
Automatic Duct
  Model Browser 8-9
  Pipes 7-6
  SINDA/FLUINT Macros 7-6

B

Background Image
  Network 4-22
Build Configuration 5-1
  Changing 9-12
  Setting Model Configuration 9-10
built-in functions 6-4

C

CAPIL 5-52
Capillary Filter 5-52
Capillary Pump
  Editing 7-3
  Introduction
  Reverseing the flow 7-3
  Rotating 180 7-3
CAPPMP. See Capillary Pump
Case Manager
  Advanced Options 9-13
  Application to multiple cases 9-14
  Defining Case Data 9-4
  Defining Cases 9-2
  Defining Output 9-10
  Introduction 3-7, 9-1
  Launching a Case 9-3
  Modifying Units 9-6
CHKVLV 5-48
Collections
  Adding and Removing Objects 4-16
  Creating 4-15
Comments
  Creating in Network 4-2
COMPPD 5-91
COMPRESS 5-80
Compressors
  Positive Displacement 5-91
    Creating
    Heritage Form A-11
  Variable Displacement
    Creating
    Heritage form A-8
Conditional Operators 6-6
Conductors
  Connecting to Pipes 7-27
  Creating 5-15
  Editing 5-16
  Generating with Nodes 5-18
  Label 4-14
  Types of Conductors 5-14
Contact C&R Technologies 2-5
CSTARTINC and CENDINC 11-4
CTLVLV 5-48

D

Defaults
  Changing 5-6, 8-5
  Control Data 9-21
  Reset Defaults 8-7
  Setting 8-5
  Setting for New Model 5-6
Disabled (Gray) 9-6
Document
  Conventions in User's Manual 2-1
  Documenting a Model 5-149
Documenting Models 5-149
  Case Manger Comments 5-150
  Network Comments 5-149
  Network Elements Comments Fields 5-150
  Thermophysical Property Comments 5-150
DPRVLV 5-49
Duct Modeling
  Automatics Duct Generation 7-6
  Disable Automatic Ducts 7-7
  Excluding Paths and Lumps 7-7
  Excluding Pipes 7-19

E

Editing
  Multiple network elements 4-11
  Network elements 4-11
  Through Model Browser 5-146, 8-2
Excel
  Post-processing Export 10-22
Exporting
  Registers 6-5
  Submodels 11-5
  Tables 5-128
Expressions
  Introduction 6-1
  Usage 6-6

F
Features 2-1
Files
  Naming convention 3-1
  Types 3-1
Filter Device 5-52
Find and Replace 3-12
Fluid Modeling Elements
  Fluid to Fluid Heat Transfer 5-111
  Fluids 5-19
  Lumps 5-23
  Paths 5-36
  Ties 5-101
Fties
  Creating 5-111
  Duplication Factors 5-113
  Editing 5-111
  Introduction 5-111
  Twinning 5-112
  Types of Fties 5-112
Functions, Built-in 6-4

G
Gauges for Post-processing 10-18
  Creating 10-19
  Customizing Scales 10-20
  Editing 10-19
Global Control Data
  Accessing 9-19
  Reset defaults 9-21
Graphical Objects
  Editing 5-148

H
Heat Source
  Lump Form Input 5-29
  Node Form Input 5-13
Heat Sources
  Source Data 9-33
Heat Transfer
  Calculated Tie 5-106
  Fluid to Fluid 5-111
  Pool Boiling Tie 5-108
  Thermal to Fluid 5-101
  User Defined Tie 5-107
Help and Support 2-3
Hover Text
  About 3-12
  Post-processing Parameters 10-13
  Setting Parameters 3-12
Hover text 3-12
HX Macro. See Pipe

I
Iface
  Creating 5-116
  Duplication Factors 5-118
  Editing 5-116
  Introduction 5-115
  Types of Ifaces 5-117
Images
  Network Background 4-22
Importing
  CSTARTINC and CENDINC 11-4
  Includes and Insert files 11-4
  Large text files 11-3
  Models from SinapsPlus 11-1
  Registers 6-5
  SINDA/FLUINT Text Files 11-3
  Submodels 11-5
  Tables 5-128
  Thermophysical Properties 5-145
Include Files 5-149
  Setting Paths 9-17
Indirect Referencing
  Data Fields 6-8
  Network Logic 5-140
Input Checking
  Automatic Macros 4-19
  Color by Layer 4-19
  Color by Material 4-18
  Highlight paths for a tie 4-21
  Missing Inputs 4-20
  Preprocessing 4-17
Interface. See Iface
Introduction 2-1

K
Keyboard Shortcuts 2-1

L
Label
  Link Toggle 4-14
  Network Objects 4-9
  Postprocessing Tools 10-20
Layers
  Color by Layer 4-19
  Creating and managing 4-2
  Deleting 4-3
  Layer Sets 4-4
  Merging layers 4-3
  Rename 4-3
  Setting Current 4-4
  SinapsPlus Import 11-2
  SINDA Import 11-3
LINE Macro. See Pipe
Links
  Creating 4-9
  Style 3-11
  Toggle Labels 4-14
Logic
  Application to multiple cases 9-14
  Case based logic 9-33
  Global 5-134
  Indirect Referencing Network Logic 5-140
  Network element based 5-137
  Submodel specific 5-135
  User defined 5-128
LOSS and LOSS2 5-47
Loss Devices 5-47
Lumps
  Changing Submodels 5-34

Clones 5-34
Creating 5-24
Editing 5-27
Label 4-9
Setting Defaults 5-35
Setting States 5-25
Wizard 5-32
Twinning 5-26, 5-29
Types of Lumps 5-23

M

Macro Commands 7-1
Duct Modeling 7-5
PIPE. See Pipe
Meniscus Modeling 5-115
Menus
Pop-up 3-10
Pull-down 3-10
Meters for Post-processing 10-18
Creating 10-19
Customizing Scales 10-20
Editing 10-19
MFRSET 5-54
Mixtures
Defining 5-22, 9-21
Setting Mass Fractions 5-35
Model
Build Configuration 5-1
Creating 5-4
Documenting 5-149
Import and Export 11-1
Input checking 4-17
Master Model definition 5-1
Model tree 8-2
Parameterizing 6-1
Setting Defaults 5-6
Submodels 5-1
Model Browser
Automatic Ducts 8-9
Editing Network Elements 8-2
Highlighting Network Elements 8-3
Model Tree 8-2
Pipes and Macros 8-4
Setting Defaults 8-5
Usage 8-1
Model Configuration
Build Statement 9-10
Changing 9-12
Mouse Operation 2-2
Multi-conductor 7-27
Creating 7-28
Editing 7-29
Post-processing 7-31

N

Network
Background Image 4-22
Collections 4-15
Copy and Paste 4-12
Creating Comments 5-149
Customizing 4-21, 5-147
Customizing Icons and Links 5-148
Drawing Objects 4-2
Editing 4-11
Element logic 5-137
Exporting 4-17
Link Definition 4-6
Object definition 4-6
Objects 4-7
Panning 4-14
Printing 4-17
View Options 4-13
Network Element Logic 5-137
Include Files 5-142
Indirect Referencing 5-140
Network Objects
Finding 8-3
Nodes
Clones 5-14
Creating 5-10
Editing 5-11
Generating with Conductors 5-18
Label 4-9
Setting Defaults 5-14
Types of Nodes 5-9
NULL 5-100

O

Object
Selection 2-3
Object Properties
Introduction 3-8
Operations Data Block
Accessing 9-22
Customizing 9-23
Options Data
Accessing 9-15
ORIFICE 5-50
Output
Binary Files 10-7
Controlling through Case Manager 9-10
Export to Excel 10-22
Frequency 10-1
Introduction 10-1
Save Files 10-7

P

Panels
Task Panels 3-2
Paths
Capil 5-52
Compressor
Positive Displacement COMPPD 5-91
Variable Displacement COMPRESS 5-80
Connecting to Twinned Lumps 5-43
Connecting to twinned tanks 5-43
Connector Definition 5-36
Connector Types 5-36
Creating 5-37
Editing 5-38
Label 4-14
Loss Devices 5-47

Null 5-100
Orifice 5-50
Pressure Regulating 5-49, 5-50
Pumps 5-57
Set Flow Devices 5-54
Setting Defaults 5-44
Tabular 5-55
Tube Defininition 5-36
Tubes and Stubes 5-44
Turbine 5-68
General Information 5-68
Twinning 5-38
Types of Paths 5-36
Valves, Check anc Control 5-48
Pipe
About 7-8
Connecting Multiple Pipes 7-24
Connecting to Network 7-24
Creating 7-9
Customizing 7-26
Discretization 7-9
Editing 7-10
Editing multiple 7-23
Exclude from aggregation 7-19
Model Browser 8-4
Moving 7-24
Orientation 7-24
Relabeling 7-25
Renumbering 7-25
Types of 7-8
Pool Boiling 5-108
Post-processing
Animate 10-21
Choked Flow 10-23
Color and Thickening Scales 10-16
Customizing Scales 5-149, 10-17
Gauges and Meters 10-18
Introduction 10-9
Network Options 10-15
Selecting Parameters 10-13
Text Options 10-22
Thermometers 10-18
Preferences
Hover Text 3-12
Panel Docking 3-15
Setting 3-11
Window Management 3-17
Preprocessor 9-3
Processor Variables
General usage 5-133
In registers and data fields 6-8
Indirect Referencing 6-8
Referencing 6-8
Properties
Fluid 5-19
Object Properties 5-146
see Thermophysical Properties
Thermophysical 5-142
PUMP 5-57
Pumps
Based on curve or map 5-57
Constant Flow Device 5-54
Heritage Form A-1
PUMP device 5-57

R

Registers
Accessing 6-2
Built-in Values 6-4
Case Specific Usage 6-7
Creating 6-2
Defining 6-2
Defining Case-based 9-18
Export and Import 6-5
Global 6-2
Groups 6-5
Introduction 6-1
Introoduction 3-4
Opening 8-7
Overrides 9-18
Showing Current Values 6-4
Usage 6-6
Reliability
Accessing 9-27
Rename
Collection 4-16
Layer 4-3
Material property 5-145
Register 6-3
Register Group 6-5
Submodel 8-2
Restart
Setting Restart Files 9-15
Robust Design (Solver combined with Reliability Engineering) 9-28

S

Save File
Location 9-4
Save Files
Creating 10-7
Scales
Customizing 10-17
Number Format Options
Post-processing
Scale Number Formats 10-17
Post-processing 10-16
Shortcuts 2-1
Solutions 9-6
Defining Solution Type 9-6
Parametric 9-9
Steady State 9-7
Transient 9-8
Solver
Accessing 9-24
Constraints 9-26, 9-30
Design Data 9-25
Logic 9-26
Procedure 9-26, 9-30
Solver Data 9-24, 9-28
Stephan Boltzmann Constant 5-5
Submodels
Activating 5-9

Adding 5-8
Changing Current 5-8
Copying 4-12
Creating 5-7
Definition 5-1
Deleting 5-8
Fluid 5-7
Import and Export 11-5
Naming rules 5-8
Rename 8-2
Thermal 5-7
Subroutines
Accessing through case manager 9-30
User defined 5-136
Support 2-3
Help File 2-3
Technical Support 2-5
Training 2-5

T

Tables 5-121
Additional Text 5-127
Case Dependent 9-18
Comment 5-127
Conversion and Scaling 5-126
Copy 5-122
Creating 5-123
Delete 5-122
Import and Export 5-128
Move 5-122
Override by Case 9-18
Structure 5-124
Table Manager 5-121
TABULAR 5-55
Task Panel
Case Manager 3-7
Docking 3-15
Floating 3-15
Hiding 3-17
Model Browser
Model Browser
Introduction 3-7
Object Properties 3-8
Thermophysical Properties 3-7
Tool Control 3-9
Text
Post-processing 10-22
Window 3-5
Thermal Modeling Elements
Conductors 5-14
Nodes 5-9
Thermometers for Post-processing 10-18
Creating 10-19
Customizing Scales 10-20
Editting 10-19
Thermophysical Properties
Creating 5-144
Importing 5-145
Introduction 3-7
Manager 5-142
Temperature Dependent 5-144
Ties
Calcuated 5-106
Centered 5-102
Creating 5-101, 5-104
Customizing Heat Transfer 5-102
Duplication Factors 5-105
Editing 5-104
Highlight Paths 4-21
Label 4-14
Lump Oriented 5-102
Path Oriented 5-102
Pool Boiling 5-108
Twinning 5-104
Types of Ties 5-101
User Described 5-107
Tool Control
Introduction 3-9
Toolbars 3-13
Customizing 3-20
Introduction 3-10
Network 3-14
Summary 3-13
TURBINE 5-68
Turbines
Creating 5-68
Heritage Form A-6
Tutorials
Introduction 12-1

U

Units
Default 5-6
Default Table 5-6
Modifying 5-5, 9-6
Setting 5-4
UPRVLV 5-49
User call routines 10-4
User Data
Accessing 9-21
User Files
Defining 9-15
Writing To a File 10-6
User Logic
Solver 9-26

V

Valves
Check Valve 5-48
Control Valve 5-48
Needle 5-50
Opening and Closing 5-50
Pressure Regulating 5-49, 5-50
VFRSET 5-54

W

Windows 3-2
Management 3-17
Tabbed 3-17
Types of Windows 3-3
Working Fluid
Include Files 9-21

Mixtures 9-21
Setting 5-19
Working Fluids
6000 Series 5-22
7000 Series 5-22
8000 Series 5-22
9000 Series 5-22
Mixtures 5-22
Setting for Submodel 8-7
User Defined 5-22
Workspace
Customizing 3-15
Overview 3-2
Working Environment 3-1

Z

Zoom
Network 4-13